POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master Degree in Computer Engineering

Master Degree Thesis

Analysis and development of Supervisory Control And Data Acquisition system for Industry 4.0

Author: Alessandro Monaco

Supervisors: Paolo Ernesto PRINETTO, Giuseppe AIRO' FARULLA

In collaboration with: *ERInformatica S.r.L.*

CEO: *Emilio Ravotti*

Project Manager: *Eng. Giovanni De Sisto*

Table of contents

1 INTRODUCTION

Supervisory Control And Data Acquisition (SCADA) systems are software systems that provide functionalities for monitoring and controlling **industrial processes**, **plants** or **other systems**.

SCADAs are placed on the top of connected hardware, typically Programmable Logic Controllers (PLCs) or other commercial hardware modules, and they interact with these devices through the exchange of messages that represent the status of systems.

During the last years, SCADA systems are becoming increasingly important thanks their capability to be integrated in different realities. Initially, they were used to supervise the status of industrial plants, but currently they have found employment in other sectors, e.g. generation and distribution of energy, steel and chemical plants, until they are used in nuclear plants.

The evolution of technologies and requirements of different applications forced the evolution and the adaptation of SCADA functionalities: every application fields have brought specific needs and goals. This led to a strong categorization of these systems.

This paper was drafted to illustrate and describe the functionalities offered by these systems and permit to know why they are gaining considerable importance in the era of Industry 4.0, considering as main topic and case of study the product Integrated Devices Open Management or IDOM.

IDOM is the SCADA system developed by the company ERInformatica S.r.L. to which I had the opportunity to join in the developing team and collaborate to the designing and installation of projects based on this product.

In the next chapters, some cases of study will be described in which IDOM acts as supervision system: for Caldirola project, the product supervises the stocking phase of sparkling wines inside autoclaves; Martini & Rossi project for the management of labelling, boxing and palletizing of wine products for Russian export; in the end, for Logico S.r.L. the SCADA is integrated as module of a MES (Manufacturing Execution System) application for elaboration of statistics about production line efficiency.

As it is introduced by the examples described previously, IDOM has potentialities and features that are based on the principle of a configurable and parametric software that allows to satisfy any requested functionalities.

2 HISTORY OF SCADA SYSTEMS

The section is intended to introduce briefly how SCADAs evolved during the last century and the motivations that allow the wide usage of this kind of systems in most of modern smart factories.

2.1 THE BIRTH OF SCADA SYSTEMS

The origin of SCADAs was in the mid-20th century, when industrial factories began to scale out in size, which causes problems in industrial organizations.

The first form of SCADA monitoring and actuations were provided by analogic sensors, relays and timers without the necessity of direct employee interactions.

However, even if the provided automation functionalities solved many problems, more issues began to arise as organizations such as scaling out persisted: relays and timers were difficult to reconfigure, fault-find and the control panels took up racks upon racks of space.

2.2 EVOLUTION OF SCADAS: 2ND GENERATION AND COMPUTER ERA – 1950S TO 1980S

In the early 1950s, computers were initially developed and used for industrial control purposes. Supervisory control began to become popular among the major utilities, oil and gas pipelines, and other industrial markets at that time. In the 1960s, telemetry was established for monitoring, which allowed for automated communications to transmit measurements and other data from remotes sites to monitoring equipment.

The birth of microprocessors and PLCs during 1970s increased enterprises' ability to monitor and control automated processes more than even before

2.3 EVOLUTION OF SCADAS: 3RD GENERATION AND NETWORKS – 1980S TO 1990S

In the 80s and 90s, SCADA continued to evolve thanks to smaller computer systems, mainframes, Local Area Networking (LAN) technology and PC-based Human-Machine Interfaces (HMI) software.

SCADA systems were able to be connected to other similar systems through proprietary LAN protocols that permit an optimized data transferring among devices and computers. Unfortunately, these systems were incapable of communicating with systems of other vendors.

2.4 EVOLUTION OF SCADAS: 4TH GENERATION AND OPEN SYSTEM MODEL – 1990S TO 2000S

In the 90s and early 2000s, SCADA systems adopted an incremental change by embracing an open system architecture and communication protocols that were not vendor-specific. This iteration of SCADA, called also networked SCADA system, took advantages of communication technologies as Ethernet. Networked SCADA systems allowed systems of heterogeneous vendors to communicate with each other, alleviating the limitations imposed by older SCADA systems, permitting also organizations to connect more devices to the network.

While SCADA systems had undergone substantial evolutionary changes, many industrial organizations continued to struggle with industrial data access from the enterprise level. By the late 1990s to the early 2000s, a technological evolution occurred: personal computing and IT technologies accelerated in development. Structured query language (SQL) databases became the standard for IT databases but were not adopted by SCADA developers. This resulted in a rift between the fields of controls and IT, and SCADA technology became antiquated over time: traditional SCADA system still use proprietary technology to handle data.

2.5 EVOLUTION OF SCADAS: 5TH GENERATION – 2000S TO DATE

Modern SCADA systems make real-time data of the plant accessible from anywhere in the world through internet connections. This access to real-time information allows governments, business and individuals to make data-driven decision about how to improve their processes. Without SCADA software, it would be extremely difficult to gather enough data for consistently well-informed decisions.

Also, most modern SCADA designer applications have Rapid Application Development (RAD) capabilities that allow users to design applications relatively easily, even if they don't have deep knowledge of software development.

The introduction of modern IT standards and practices such as SQL and web-based applications into SCADA software has greatly improved the efficiency, security, productivity and reliability of SCADA systems.

SCADA software that use the potentialities of SQL databases provide huge advantages over antiquated SCADA software. The main advantage of using SQL databases with a SCADA system is that it makes it easier to integrate into existing Manufacturing Execution Systems (MES) and or Enterprise Resource Planning (ERP) systems, allowing data to flow seamlessly through the whole organization.

Historical data from a SCADA system can also be logged inside the SQL database, which allows data analysis through data trending.

3 ARCHITECTURE OF A SCADA SYSTEM

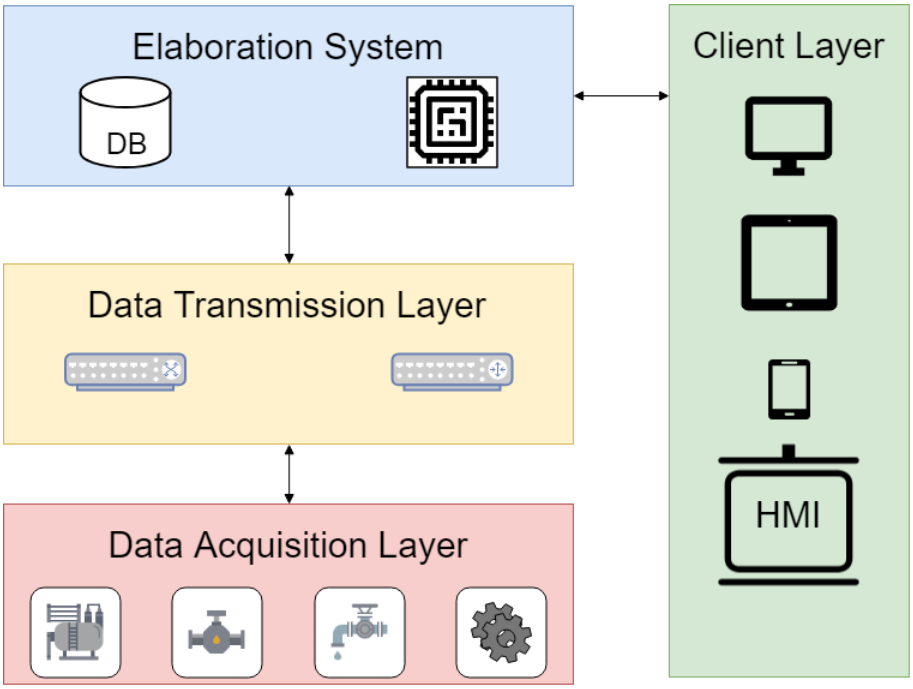

Figure 3.1 SCADA layered architecture

The figure above offers a general overview about the components that compose a generic SCADA application. This model is a valid solution that represent the future of this kind of software system. It is a layered architecture composed of different layers:

- **Data Acquisition** layer, the layer responsible to the acquisition of data from the surrounding world
- **Data transmission** layer, the layer responsible to the transfer of data from peripheral data acquisition devices to the elaboration system
- **Elaboration system**, it is the core of the SCADA
- **Client** layer, it is the layer that acts as a bridge between the SCADA system and operators

3.1 DATA ACQUISITION LAYER

The Data Acquisition layer is the part of a SCADA application that interfaces with the surrounding world.

The main components of this layer are the data acquisition devices that constitute the means by which a SCADA system could know the status of industrial plants and processes.

These devices can perform the following operations:

- Collecting data from the monitored process using sensors, cameras or any other kind of input and converting them into a manageable format, typically digital or analogic measurements.
- Actuating operations on the controlled process using actuators, procedures or any other kind of output, e.g. activating or deactivating a conveyor.

Sensors have the purpose to translate external physical quantities into electrical signals that are readable for controllers, mainly PLCs or data acquisition boards.

Actuators have the opposite purpose: they receive commands as electrical signals then they perform the requested operations in the surrounding world, e.g. turn on a lamp, start an engine or move mechanical parts.

These devices are fundamental for a plant because they permit to know what happening during the working phase and perform tasks on it. For this reason, they must react in real time constraints in order to avoid misbehaviours of the entire process. A generic device must perform during this short period this set of operations:

1. Collecting the physical/electrical signals
2. Converting them into a manageable format
3. Holding the measurement/performing the requested processes

The conversion strategies have the crucial role in terms of information quality and impact on performance of acquisition devices, because performing less operations guarantees great benefits on the whole application.

As any system, devices are subjected to usage limitations that must be analysed at the beginning of the projects during the project specification definition. This fact permits to avoid future issues and misbehaviours of the entire SCADA application.

3.2 DATA TRANSMISSION LAYER

The Data Transmission Layer is the part of a SCADA application that provides the functionalities of data transmission between peripheral data acquisition devices and the central elaboration system.

The definition of interfaces among controlled processes, devices, elaboration system and sub-systems is a crucial phase that allows to know how every actor can transmit data to others.

The best practice asks to answer the following questions according the wanted system:

- Speed
- Security
- Reliability

- Availability
- Intelligibility

Obviously, it is impossible to deploy a system without meeting the need to do compromises.

3.2.1 Speed

The data exchanging speed analysing is necessary to make efficient a SCADA system: the analysis considers the dynamic characteristic of the whole process in order to deploy a compliant communication channel sufficiently fast to allow the whole acquisition, transmission, processing, re-transmission, actuation at such times to make control actions effective.

Constraints are imposed by the controlled process and they could be extremely stringent to force the usage of distributed system. An example is the management of the gas transport systems that need high-speed communication channels and they often require peripheral structures that can actuates controlling policies autonomously.

Wide area systems suffer most the speed limitations that are imposed by the current technologies due to the impossibility to build dedicated infrastructures instead of using a general purpose one.

The communication about elaboration system and human-machine interfaces or HMIs must be sufficiently quick to permit the correct perception of the evolution of the process and an effective reaction of the system at any human interactions.

3.2.2 Security

The security of elaboration and communication systems is relevant when they are exposed to intrusions that could be potentially dangerous. The security is not intended for “bad faith” of people, but also to bad usage and errors of operators.

The security management should consider the point of view of communications among elaboration system and peripheral devices and interactions with external systems that could influence the behaviour of the SCADA system and so the controlled processes.

The common resolution is the restriction of areas and activities of employees. This restriction could be physical (e.g. different rooms) or logical (e.g. granted access with the insertion of credential).

The realization of security policies is usually managed with levity because this kind of systems are closed and insulated by the external world. Only during the last years this approach is changing due to the availability of small and cheap communication systems that support network protocols like UDP or TCP.

3.2.3 Reliability

The transmission of data is a process that is characterized by the presence of phenomena that could affect the integrity of information and so a real risk during the evaluation of the status of the process.

In a controlling system the possibility to validate the received information is not possible due to the low evolution time, so it is demanded to transmission systems.

Modern technologies and protocols adopt controlling algorithms that allow identifying failures over heterogeneous infrastructures that affect the network properties.

The assurance to receive data flows without the presence of error involves the presence of mechanism for error identifying, re-transmission of the correct version of data and order management of information unit inside the flow.

The opposite solution is to redirect the data flow without checking its integrity then verifying it at the receiver side. This kind of solution is very effective for transmission means that guarantee a very low bit error rate (for example optic fibers guarantee a bit error rate less than 10^-9) or small infrastructures.

3.2.4 Availability

In a SCADA system, the analysing of effects due to disservices is important. Generally, a supervisory system is characterized by a direct relation between communication services availability among devices and controlling activities.

Usually, the internal communication system of a controlling system should adopt continuity strategies that allow to guarantee the service available at the least the requests of the whole system.

This behaviour happens for most of the systems that implement controlling tasks, but for the only monitoring ones these constraints are not imposed because they are implemented with discontinues nature communications: the peripheral devices could store temporally data then send in bulk them to the central system and clean the internal storage.

If the system adopts controlling policies, it needs a continuously updates of the process status and the availability of communication channel for actuations.

In order to ensure the satisfaction of these constraints, it is needed the adoption of technologies and protocols that provides services with these features and set up of an infrastructure that are able to bridge eventually failures and faults, for example with redundancy strategies.

3.2.5 Intelligibility

The intelligibility of the information is a fundamental element for the realizations of system that should have high evolution capability for functionalities and the interaction with external systems.

During the definition of interfaces is a good practice referring to standard models because they permit to guarantee the reliability of adopted solutions and a longer life expectancy.

3.3 ELABORATION SYSTEM

The elaboration system is the core module of any SCADA applications.

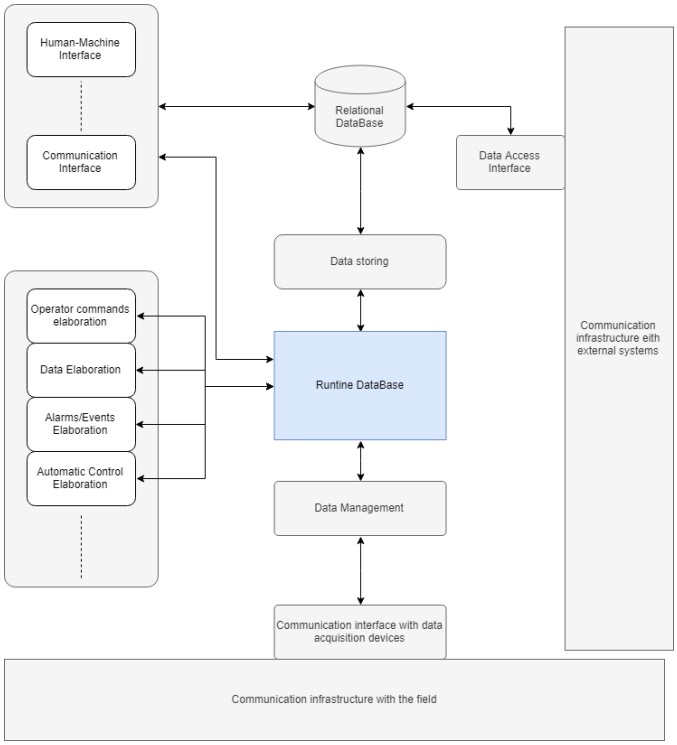

Figure 3.2 Model of elaboration system

The proposed model (figure 3.2) illustrates the common adopted solution of elaboration structures, but it represents also a functional architecture that should remain valid in view of the evolutions to which the SCADA systems seem destined and, therefore, a tool for the presentation of typical features of this kind of applications.

The common architecture is a layered one which is composed of:

- **Data management** module
- **Data elaboration** module
- **Data sharing** module

3.3.1 Data Management Module

The key element of a supervision and control system is the data management module.

Its role is to receive data from the data acquisition devices (or *field*) that represent the controlled process status and try to standardize in a manageable convention defined by the application.

After this process, data flow to a collection point called *database,* or briefly *DB*, that acts as source and destination of all necessary information for any module of the application.

In this kind of systems, which requires a continuously access to data, the database must guarantee high performance in terms of I/O requests. For this reason, the used database appertains to the category of *Runtime DB*: it is a technology solution that permits to collect data that represent exclusively the working status of monitored processes. This memory area allows a fast accessing to most used data.

This kind of data management modules permits to satisfy all fundamental requests and functionalities of SCADA systems, but it does not allow to gain knowledge about the evolution of processes.

For this reason, the application must provide functionalities that have characteristics of a relational database: the properties and the functionalities that favour the choice this kind of DBs are:

- The intelligibility of information
- The availability of tools for data accessing from application side
- The efficient management of huge amount of data

The integration of relational databases can't replace the functionalities offered by runtime DBs because the I/O operation are too slower for real time processing, but they can used for all operations that interact with the history of the monitored processes which do not requires high performances, contrary the real time analysis.

3.3.2 Data Elaboration Module

The controlling system provides a tool for the information exchange process that is able to receive the status of the plant and send commands for controlling the evolution of it.

The huge quantity of these data and the complexity of the controlled process make impossible a precisely analysis to operators. This fact suggests deploying automatic procedures that will be responsible about the generation of commands and aggregate information.

The management of railway traffic could be an optimal example that permits to understand the complexity and the importance of automatic elaborations: in order to know if a route is available to accept a train, it is necessary execute the following checks:

- The railway is correctly aligned

- The route is not occupied by other trains
- The route is under maintenance activities
- The signage is correctly configured to stop any incoming trains

Without the help of automatic procedure that are provided by an elaboration system, the operator has to check any conditions manually, with the consequence to increase the risk to lose the focus on the goal (checking the availability of a route)

The realization of calculation processes responds to such needs:

- Generation of summarized representations of process status
- Generation of reports about anomalies on process
- Interpretation of operators' command and actuation of punctual procedures
- Generation of aggregated information that represent the evolution tendency of processes
- Execution of automatic controls and procedures

The last point of the list requires further clarifications: in case of the dynamic of processes does not allow a timely intervention from operators or SCADA elaboration system, it is usual deploying distributed automation elements that execute autonomously actuations and controls.

3.3.3 Data Sharing Module

In the last chapters it is described a system that guarantees operability and efficiency in terms of monitoring and controlling of industrial processes. However, it presents limits in data access source: only the planned HMIs can interact with them and only for the suitable functionalities.

The activities that are collected by the SCADA are commonly subjected to interests that are outside of operators' command set, but they belong to analysis adopted for consulting or prevention.

For this reason, a SCADA system must make available all collected data from external services, for example with the direct access to them.

An alternative solution could be the realization of internal functionalities or tools that produce the required aggregated information. This solution is usually avoided because it implies a forcing in the nature of this kind of application.

In conclusion, it is required to deploy a sharing system that permits operators and external services to retrieve and elaborate the needed information according their purposes without the intervention of the SCADA system. What is expected by sharing systems is the possibility to reach and interact with a database, specifically a relational database, and elaborate the retrieved information according what is demanded.

3.4 CLIENT LAYER

The client layer is the part of SCADA that realize the functionalities of representation of collected data on displays, called Human-Machine Interfaces or briefly HMIs, which are accessible by operators.

HMI designers have the responsibility to represent the controlled process through simple and intuitive graphic interfaces that allow making the system “transparent” to operators. Taking care of the usability and user experience of the SCADA, operators should be facilitated to execute their tasks and will gain feeling with the whole system, limiting the stress level at critical moments.

The main functionalities of HMIs include:

- Representation of data
- Management of user commands

3.4.1 Representation of data

The representation of data allows operators to know the status of the monitored process by displaying information using textual or graphical components, for example charts, gauges, tables or indicators.

3.4.2 Command management

A fundamental part of any SCADA systems consists of the possibility to send command to the controlled process. They are divided in two groups:

- Automatic command: this kind of commands represents any tasks that is autonomously elaborate by the elaboration system according the status of controlled processes
- Manual command or User command: this kind of commands represents any tasks that could be run by operators.

4 IDOM

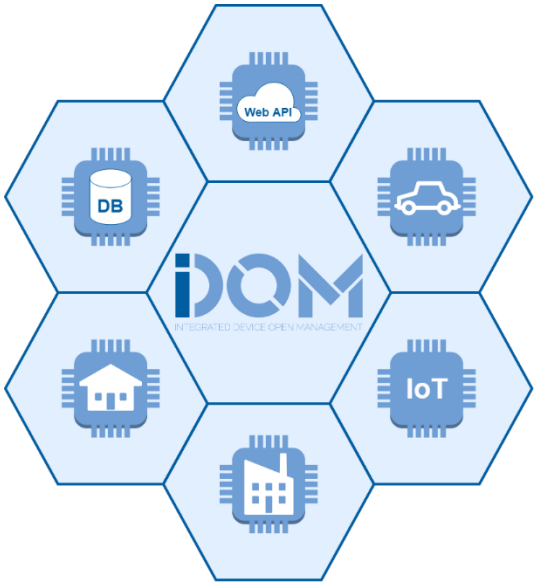

Figure 4.1 Application fields of IDOM

Integrated Devices Open Management or IDOM is an advanced software system, developed by ERInformatica S.r.L., that implies the functionalities of SCADA applications.

The project was born in order to meet the needs of automation companies and factories that want deploying new production lines that involve innovative technologies according the concept of smart factories and Industry 4.0, but the product could be integrated in various environments and applications due to product interfaces, as the figure 4.1 illustrates.

It is a Java- and Spring-based software that follows the methodologies and structures of enterprise software.

Its jobs could be identified in two main groups:

- Traditional functionalities:
  - Monitoring
  - Controlling
  - Analysing
- Advanced functionalities:
  - Preventive maintenance scheduling
  - Video content analysing
  - Responsive UI and dashboard configurations
  - Geolocation functionalities
  - Remote notification and command engine
  - Web services and API for system inquirying

The core features of the product are its great configuration capability and integrability, which allow to make IDOM easily embeddable in plants with the unique constraint of communication through network interfaces over buses like RS-232 or RS-485.

IDOM is a platform independent java-based web application whose UIs are accessible on any web browser client.

4.1 IDOM APPLICATION ARCHITECTURE

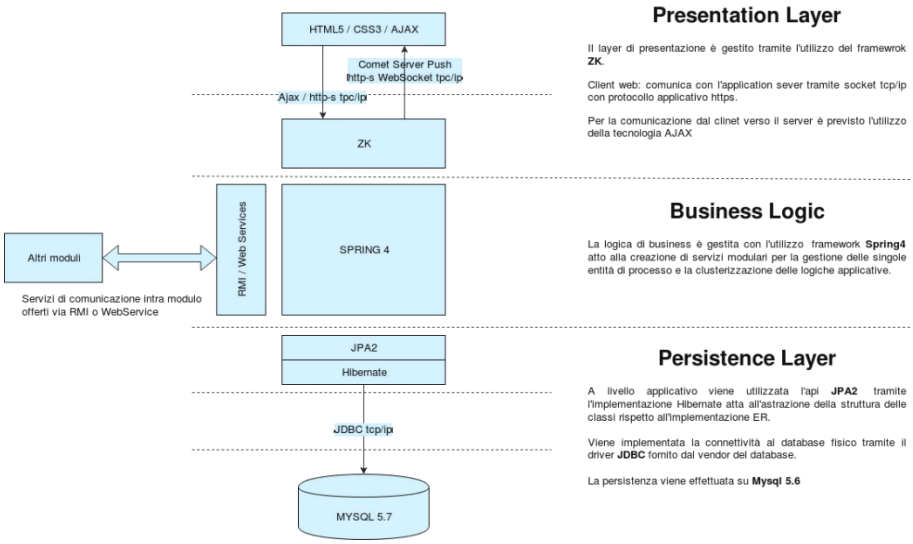

Figure 4.2 IDOM architecture

The figure 4.2 illustrates the software architecture of a generic IDOM instance.

It is a layered one in which exists three layers:

- Presentation layer
  - ZK 7 framework
  - Ajax and web sockets for communicating with business logic layer
- Persistence layer
  - JPA2 interface using Hibernate
  - JDBC driver
  - RDBMS MySQL 5.7
- Business logic layer
  - Web services
  - Spring 4 Core
  - Driver communication library

4.1.1 Presentation Layer

The presentation layer is the part of the software that manages the interactions between IDOM and users through web pages that are designed by the utilization of tools and facilities provided by ZK, an open source Ajax web application framework.

4.1.1.1 ZK Framework - from ZK documentation

"ZK is a component-based UI framework that enables you to build Rich Internet Application (RIA) and mobile applications without having to learn JavaScript or AJAX. You can build highly-interactive and responsive AJAX web applications in pure Java. ZK provides hundreds of components which are designed for various purposes, some for displaying large amount of data and some for user input. We can easily create components in an XML-formatted language, ZUL.

All user actions on a page such as clicking and typing can be easily handled in a Controller. You can manipulate components to respond to users' action in a Controller and the changes you made will reflect to browsers automatically. You don't need to care about communication details between browsers and servers, ZK will handle AJAX details for you. In addition to manipulating components directly i.e. MVC (Model-View-Controller) approach, ZK also supports MVVM (Model-View-ViewModel) approach which decouples the Controller and View. These two approaches are mutually interchangeable, and you can choose one of them upon your architectural consideration.

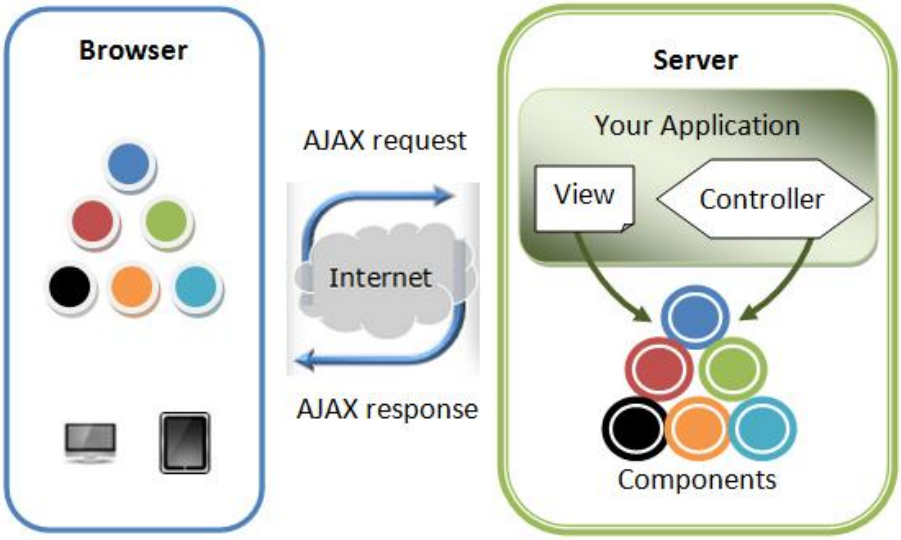

Figure 4.3 Simplified ZK Architecture

Above image [figure 4.3] is a simplified ZK architecture. When a browser visits a page of a ZK application, ZK creates components written in ZUL and renders them on the browser. You can manipulate components by your application's Controller to implement UI presentation

logic. All changes you made on components will automatically reflect on users' browser and ZK handles underlying communication for you.

ZK application developed in a server-centric way can easily access Java EE technology stack and integrate many great third-party Java frameworks like Spring or Hibernate. Moreover, ZK also supports client-centric development that allows you to customize visual effect or handle user actions at the client side."

4.1.1.2 Kinds of User Interfaces

IDOM user interfaces are grouped in different groups that now are described.

4.1.1.2.1 Administration User Interfaces

The administration user interfaces are a collection of interfaces that are deployed for the configuration of the system. They are accessible only by employees of ER Informatica with a high grant level, defined also "*Power users*", because they let them to configure the following details of the system:

- **System properties**, e.g. the thread pool sizes.
- **Scheduler** configurations
- **User permissions**, **profiles** and **groups**
- **Drivers** configurations

4.1.1.2.2 Back-End Configuration User Interfaces

The back-end configuration UIs stands for the collection of web UIs that let users to configure the major entities of the application and plants:

- **Event categories**
- **Devices**
- Registered **Users**
- **Data retention policies** of data
- Manual/Automatic **Scenarios**

These UIs are accessible to users that has high permission level, called also *"Configurators"*.

4.1.1.2.3 Front-End configuration User Interfaces

This kind of user interfaces is exploited to configure the customizable front-end of IDOM.

Users that could access to these UIs are known as *"Dashboard configurators"*.

This section will be described with details at the dedicated chapter 4.1.11.

4.1.1.2.4 Dashboards and Synoptics User Interfaces

The last collection of user interfaces is called **Dashboard** and **Synoptics** UIs.

They represent those web pages or UIs that are customized from the users that appertain to the group of “Dashboard configurators”.

Users can access to any dashboard or synoptics to which they are registered/granted.

This section will be described with details at the dedicated chapter 4.1.11.

4.1.2 Persistence Layer

The persistence layer is the part of IDOM that performs the jobs of storing and inquiring of collected data from/to database.

It is implemented using JPA2 (Java Persistence API v. 2) interfaces, in details with Hibernate as Object Relational Mapping framework (ORM), that define the interfaces of data accessing from Java application to database tables.

Java Persistence API is a Java specification that define which facilities are requested for managing relational data in Java application. It could be identified according four areas:

- Java Persistence API
- Java Persitence Query Language or JPQL
- Java Persistence Criteria API
- Object relational mapping metadata or ORM

As introduced above, IDOM uses Hibernate as ORM framework. It is the most used opensource ORM that provides implementations of JPA interfaces. The purpose of ORMs is the management of application entities automatically and transparent for users, writing their properties into the correlated tables of relational database.

Java entities are mapped to database tables by the implementation and configuration of XML configuration file – persistence.xml – or by using provided Java annotations which are used by Hibernate in order to generate and maintain the database schema.

The persistence layer is grouped in two different modules in base of their purposes and tasks:

- *Historical data management*: this group is intended to satisfy all requests that interact directly with the database in order to retrieve historical events, e.g. the temperature trend.
- *Realtime data management*: due to the continuously access to the database from IDOM, it is required to access real time events quickly. For this reason, IDOM provides a runtime database that contains only the current status of the controlled systems. Thank this component, inquiries, elaborations and controls are speed up.

4.1.3 Business logic layer

The business logic layer is the core layer of IDOM and it manages all provided services and functionalities involved in any installation of the product.

The core of this layer is composed of different modules and engines that are illustrated in the following picture:

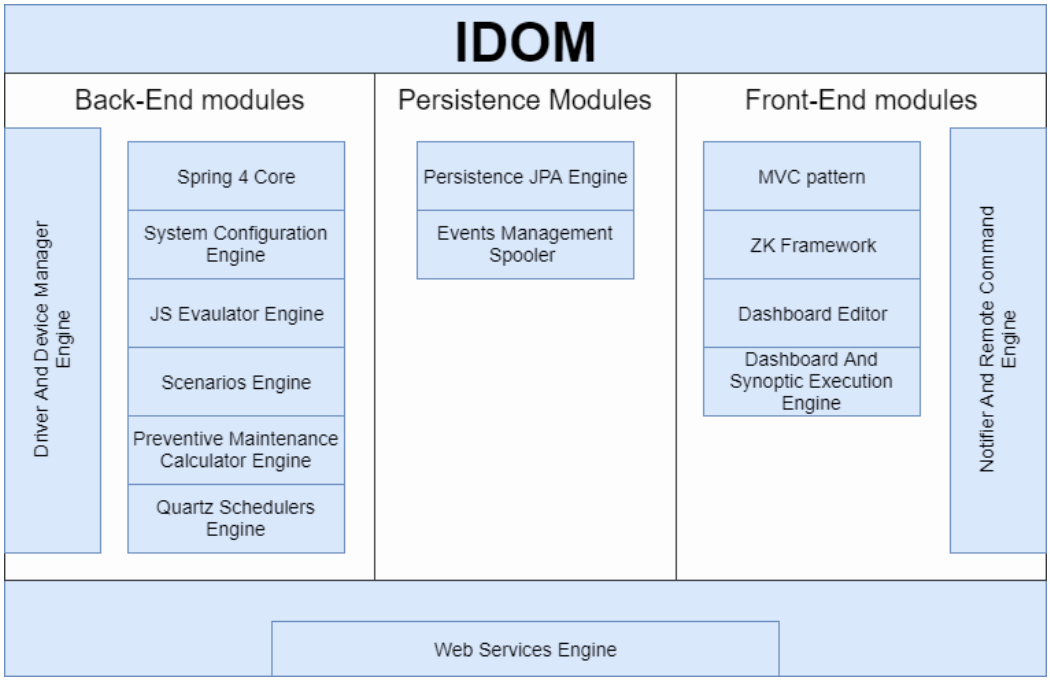

Figure 4.4 IDOM modules and engines

The next sections are intended to describe some of these modules.

4.1.3.1 Spring 4 Core

Spring is an open-source framework for the development of Java applications which allows building applications using POJOs (Plain Old Java Objects) with enterprise services according the JAVA SE and EE programming model.

Spring provides the commonest functionalities of enterprise world (e.g. Data Access Objects, Object-Relational Mapping and web) and promotes the interface-oriented programming in order to reduce the dependability on framework APIs.

The architecture of Spring is based on a layered one that permits the integration of the commonest technologies. This kind of architecture permits to choose those modules that will be integrated according the requirements of the application.

The main feature of the framework is the *Inversion of Control* or IoC: its purpose is to produce extremely decoupled e light code. The main idea of IoC is that objects expose requests and dependencies that will be provided by the IoC container.

Inversion of Control could be named also with the term *Dependency Injection* (DI).

The injection of dependencies could be implemented in three different ways:

- *Constructors*: dependencies are injected through the constructor method of classes.
- *Setters*: dependencies are injected through setters
- *Abstract methods* (not natively supported in Spring)

4.1.3.2 Pattern Model-View-Controller

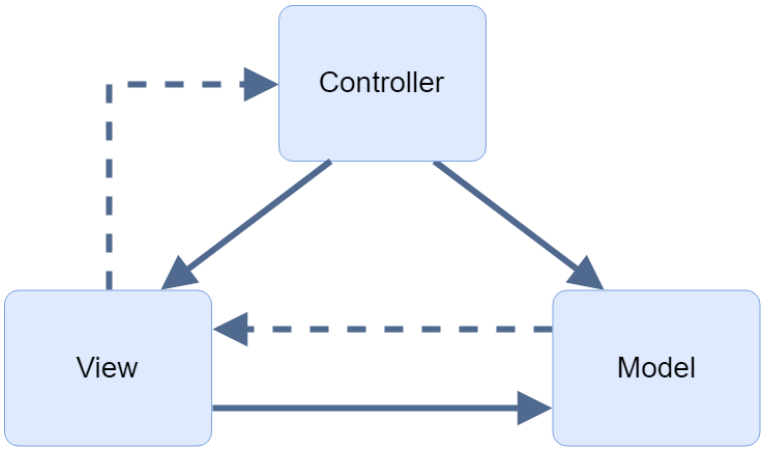

Figure 4.5 MVC Pattern diagram

The Model View Controller or MVC pattern is an architectural pattern for software development that allows to divide business logics, view logics and modelling in different modules.

The architecture is composed of three components:

- **Model**: it is the core of MVC pattern that defines the behaviour of the application in terms of application domain, independently about user interfaces. This component manages directly data and logics of the application.
- **View**: it is the component of MVC pattern that shows models and data according the requirements, e.g. diagrams or tables.
- **Controller**: it is the component of the MVC pattern that elaborates and converts user inputs into commands for views or models.

4.1.3.2.1 Advantages

- *Simultaneous development*: multiple developers can work simultaneously on the models, controllers and views.

- *High cohesion*: MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- *Low coupling*: the very nature of the MVC framework is such that there is low coupling among models, views or controllers
- *Ease of modification*: because of the separation of responsibilities, development and modification integration is easier
- *Multiple views for a model*: models could own multiple views

4.1.3.2.2 Disadvantages

- *Code navigability*: the framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria of MVC.
- *Multi-artefact consistency*: decomposing a feature into three artefacts causes scattering. Thus, requiring developers to maintain the consistency of multiple representations at once.
- *Pronounced learning curve*: knowledge on multiple technologies becomes the norm. Developers using MVC need to be skilled in multiple technologies.

4.1.3.3 Driver And Device Management Engine

One of the aims of IDOM is the possibility to install instances of the SCADA in a wide range of application fields. To provide this feature, the product integrates internally an engine called *Driver And Device Management Engine*.

The module permits to manage any drivers and devices that IDOM will use by the configuration of appropriate properties, e.g. the IP address of remote devices or parametric behaviours of the drivers.

The module is composed of two sub-modules that will be described:

- Driver management, the component that manages drivers
- Device management, the component that manages devices
- IDOM-Device communication interface, that defines the permitted operation between IDOM and a generic driver/device

4.1.3.3.1 Driver management and *IDriver* interface

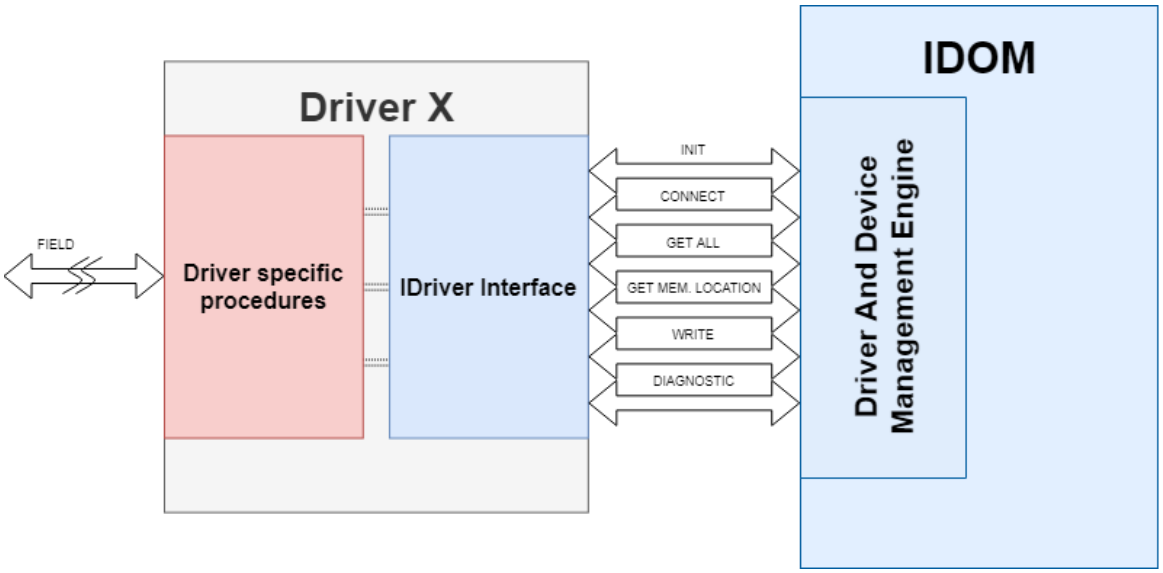

Figure 4.6 IDOM and IDriver interface

The main issue encountered was the management of the huge differences between device types in terms of communication protocol, available services and permitted operations. The solution adopted in IDOM to overcome this obstacle consists of the definition of a common interface, called *IDriver*, that permits to abstract driver/device implementations: it declares the set of operations and tasks that any driver implementations must have in order to consider it integrable in IDOM. These operations are:

1. Initializing the driver for i-th device
2. Connecting to device
3. Getting all memory locations from devices
4. Getting punctually a memory location from devices
5. Writing a memory location in a device, excepting for read-only data sources (e.g. files or databases)
6. Performing diagnostic procedures, e.g. connectivity tests

Once the driver is installed in IDOM, it can be used to connect associated devices.

Drivers are dynamically instantiated in IDOM context using the features of the Java ClassLoader that permits to load dynamically the bytecode of drivers and create instances of them through the available APIs. This solution generates a strong abstraction about the internal procedures of driver implementations.

4.1.3.3.2 Device management and IDevice interface

Likely what happens for communication drivers, also devices require to implement a common abstract interface in order to be configured in IDOM. This interface is called *IDevice* and it allows developers and configurators to abstract properties and functionalities that characterize devices, mapping them inside IDOM configurations.

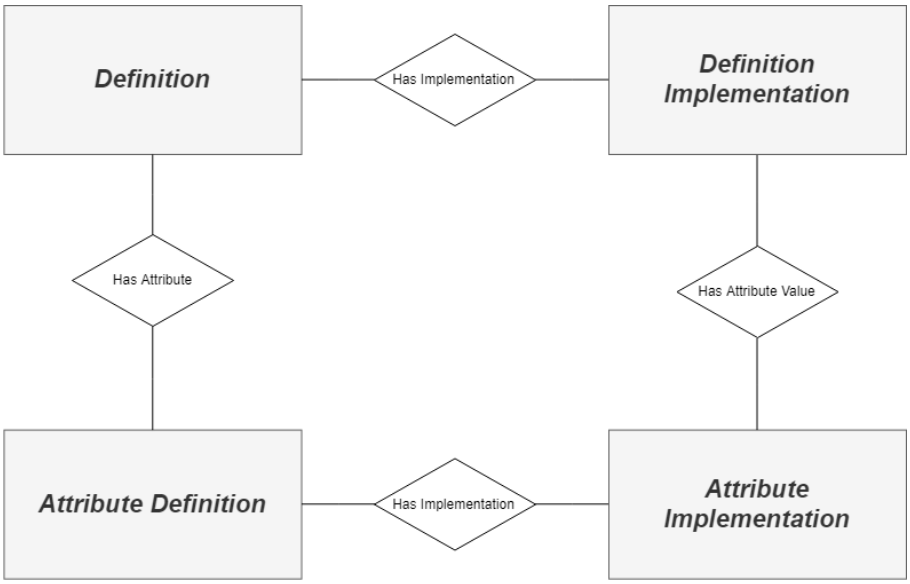

Figure 4.7 Entity Relationship diagram of the flexible data structure integrated in IDOM

A flexible data modelling is implemented in the SCADA system that acts to the definition of:

- **Metadata**: they define how real data are modelled in the system
- **Data**: they represent the effective values of a specific definition through the metadata described above

Thanks to the metadata & data approach, it is possible define any structures or reality representations with an approach ideally close to Object Oriented Programming principles.

It is evident the advantages of the chosen approach, because, which the usage of what is introduced above, it is possible define a heterogeneous set and potentially unlimited of:

- **Drivers** for communicating with devices, e.g. it is possible implementing a collection of them for integrating the main PLC vendors that will be used transparently by the business logic layer
- **Devices**, with the capability to characterize any kinds of them, starting by a simple sensor with a single output to complex machines that involve thousands of I/O signals

Analysing the implemented solution in greater detail it is possible define the following entities, described in the figure 4.7:

- **Definition**: it represents the definition of a class of objects
- **Attribute Definition**: it represents an attribute of the related Definition
- **Definition Implementation** or Implementation: it represents an instance of a specific Definition
- **Attribute Implementation**: it represents the valuation of the i-th attribute related to a specific instance of a specific object class

4.1.3.3.3 IDOM-Device communication: reading procedure

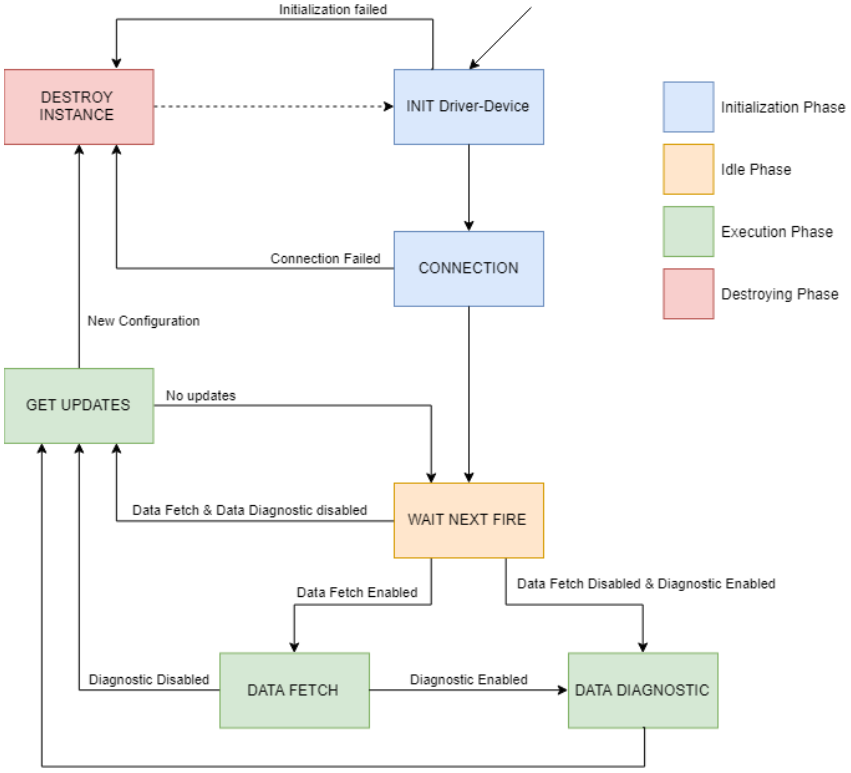

Figure 4.8 Reading procedure cycle

The diagram 4.8 represents the life cycles of a driver-device instance during the reading procedure. It consists of a final state machine that evolves in its internal state continuously according the configuration adopted by drivers and devices.

The states are:

1. **INIT Driver-Device**: Initialize the communication parameters between IDOM and device
2. **Connection**: IDOM tries to connect to device
3. **Wait Next Fire**: Wait the scheduling from IDOM
4. **Data Fetch**: Executing the complete fetching of data from device
5. **Data Diagnostic**: Perform diagnostic procedure on device / Fetch diagnostic data from device
6. **Get Updates**: Check and retrieve any configuration changes for driver or device
7. **Destroy instance**: Stop and destroy the driver-device instance, if required by the system.

4.1.3.3.4 IDOM-Device communication: writing procedure

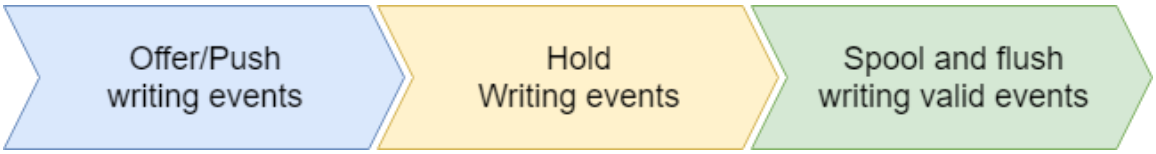

Figure 4.9 Writing procedure

The writing procedure to remote memory locations is an extremely delicate task that any SCADAs have to achieve using clever strategies.

The main issue is the synchronization and the serialization of writing requests in multithreading and distributed environments like IDOM: it could happen that more services want setting remote memory locations or parameters in order to achieve their interactions.

The idea at the base of the solution provided by the product consists of the implementation of a tasks spooler that achieves the priority assignments using a pool of Last Input First Output event buses, one for each memory locations that have at the least one writing operation to be managed.

The adopted solution ensures that only the last task for each memory locations, which represents the unique writing operation that could be executed.

How it is possible understanding, the solution permits the correct setting of memory locations, but it does not manage the eventually failures caused by rejections.

The drawbacks of this solution were analysed deeply by developers and the obtained reports had demonstrated that they can be avoided through a careful configuration of actors and tasks involved.

4.1.3.3.5 Drivers and Device Synchronization issue

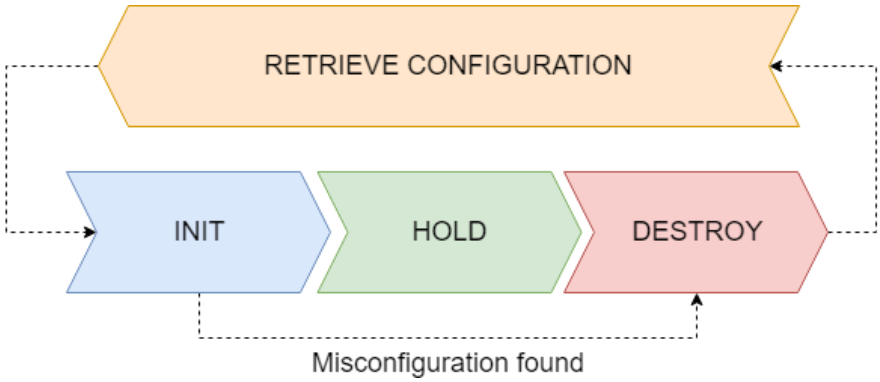

Figure 4.10 Life Cycle of a driver/device instance in IDOM

The capability of IDOM to create, destroy or reload instances of drivers and devices using Java ClassLoader APIs led to a considerable increase of performances of the product, however it led also to the issue of configurations synchronization and instances pollution that involves a high usage and waste of HW resources of the server in which the SCADA is installed.

For this reason, IDOM provides internally to the engine an effective and efficient engineering of the synchronization strategy: the solution consists of the implementation of a listener that collects all active drivers and devices that IDOM instantiates according the configurations of the SCADA and, in case of configuration updating, it destroys those instances that are no longer valid.

The diagram 4.10 illustrates the life cycle of drivers and devices inside the products:

0. **Retrieve Configuration**: IDOM offers to the listener the most recent configuration of the driver/device
1. **Init**: the listener instantiates the driver/device configuration
2. **Hold**: the instance, which is represented by a thread, is cached internally the service, so it is easily accessible by IDOM according direct access strategies based on a hash map in order to execute the scheduled operations, e.g. reading or writing memory locations
3. **Destroy**: when an instance is marked as deprecated or invalid, the listener destroys the associated instance and thread in order to avoid the pollution and resource wasting. Meanwhile the destroying process is running, the service will instantiate the latest configuration related to the driver/device in order to guarantee the continuity of the service.

4.1.3.3.6 Advantages

This management strategy adopted in this solution allows IDOM to improve resources management of the server in which it is installed and, consequentially, to ensure the reliability of the SCADA.

4.1.3.3.7 Disadvantages

The benefits gained by the integration of the Driver and Device Management Engine make IDOM extremely flexible in terms of management of the field.

The main drawback consists of the required resources for the synchronization of drivers/devices configurations internally the application, but the involved efforts are worth and permits to increase considerably the performance of the whole SCADA.

4.1.3.4 JavaScript Evaluator Engine

The *JavaScript Evaluator Engine* is an internal engine of IDOM that allows the execution of JS scripts in any part of the Java code of the SCADA. It is one of the core modules of the product that makes IDOM flexible and integrable in a wide set of application fields.

The main uses of this engine are:

- Event pre- and post-persisting procedures
- Custom reading conditions
- Execution Logics or Scenarios executions
- Dashboards and Synoptics interactions

4.1.3.4.1 Advantages

The integration of JS Evaluator Engine permits IDOM to be flexible and usable in a wide set of application fields. It is possible deploying customized controls and functionalities that otherwise require long time in terms of analysis, configuration and development.

4.1.3.4.2 Disadvantages

The integration of the JS evaluator engine took great advantages in terms of flexibility and usability of the SCADA system, as described previously.

However, this solution implies few drawbacks that IDOM configurators have to manage during the designing and configuration of the product. These drawbacks are listed below:

- The integration of JS requires to IDOM configurators to learn at the least the basis of this scripting language in order to configure functionalities that requires the scripting language
- SCADAs are multithreading systems that can execute multiple tasks at time. This characteristic took great advantages in terms of performance, but it took also drawbacks in term of data consistency or data synchronizing. The evaluation of JS scripts could be taken in different instances and, a cause of the evolution dynamic

of monitored processes, they could get different values for the same event also if them will be executed in parallel.
- The correctness of JS scripts is demanded to configurators

4.1.4 Events Persisting Spooler

As said at the chapter 3, the interaction with a relational database represents the bottle neck of this kind of systems, but generally for any other software systems.

The huge amount of I/O requests to/from DB could undermine extremely the performance of the whole system. To avoid this issue, IDOM implements a module that is called Event Persisting Spooler: it consists of a spooler that enqueues any events according a Last Input First Output data structure that has the jobs to spool events and persist them inside the relation database (in this cases MySQL).

The idea at the base of this spooler is that more recent data are, greater is them priority.

The Events Persisting Spooler acts also as runtime DB because it stores internally the last data of each events that were fetched from devices. This cache permits to deploy a direct access (using a hash map) to the last value of each event categories and, so, reduce the latency of I/Os.

4.1.5 Programmable Execution Logics or Scenarios

Execution logics or Scenarios are the elements of IDOM that allow to execute operations and procedures on the monitored system.

Their configurations are demanded to back-end configurators that specified which data IDOM has to exchange to the monitored system in order to execute the issued procedures.

Scenarios are divided into different main groups:

- Manual scenarios or User commands
- Automatic scenarios

4.1.5.1 Manual scenarios or User commands

This kind of execution logics defines all procedures that IDOM will perform any time they are requested from external events, like pressing a button on a dashboard or third-party application requests.

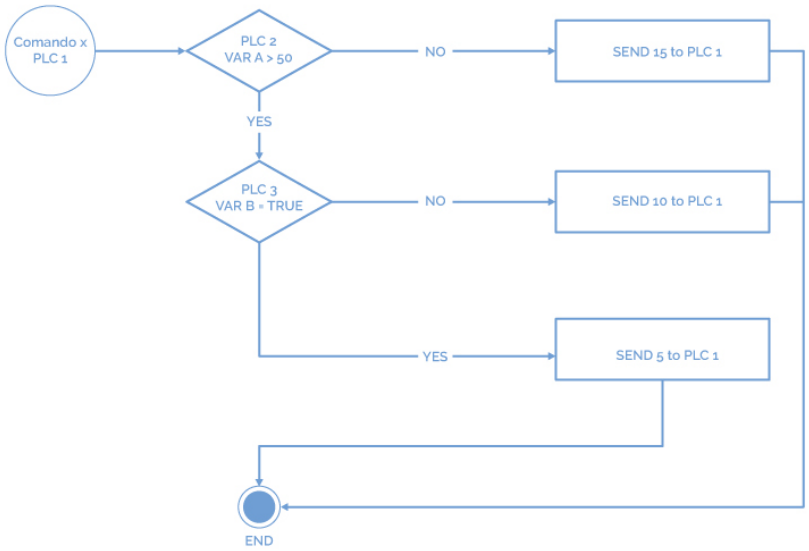

Figure 4.11 Manual Scenario diagram

The flow diagram illustrates an example of User Command that could be started by any users: the procedure permits to send a value to PLC1, which could be a conveyor controller or a heater for example, in order to set the ripple per minute or temperature set point according the values of other PLCs.

In this example it is possible to summarise the procedure using the following C++ code:

```
void scenario(){
    /**
    * GET ALL DEVICES
    */
    Device plc1 = DeviceManager.get("plc1");
    Device plc2 = DeviceManager.get("plc2");
    Device plc3 = DeviceManager.get("plc3");
    /**
    * GET ALL VALUE FROM DEVICES
    */
    int plc2_a = (int) plc2.get("A");
    bool plc3_b = (bool) plc3.get("B");

    /**
    * SCENARIO
    */
    int sendValue;
    if(plc2_a > 10){
        if(plc3_b){
            sendValue = 5;
        } else {
            sendValue = 10;
        }
    } else {
        sendValue = 15;
    }

    /**
    * SEND VALUE TO DEVICE (REMOTE MEMORY LOCATION IS CALLED "C" IN THIS EXAMPLE)
    */
    plc1.set("C", sendValue)
}
```

Figure 4.12 C++-like representation of manual scenarios

4.1.5.2 Automatic scenarios

This kind of executable logics defines the automatic procedures that IDOM will perform without the need of human or external interventions.

These scenarios are implemented using a configurable static scheduling mechanism based on the instantiation of threads.

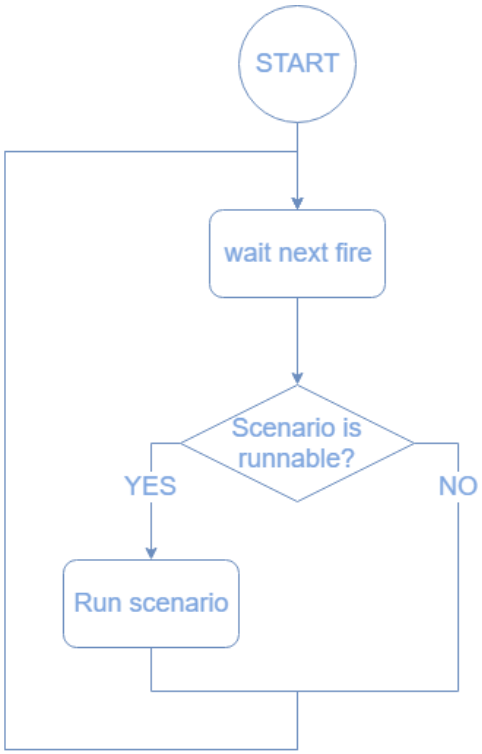

Figure 4.13 Automatic scenario diagram

The diagram illustrates the steps that are executed in automatic scenarios. It may notice that only one schedules at time can be run.

```
void automatic_scenario(){

    while (true){
        long current_timestamp = <<now>>;

        if(is_runnable())
            scenario();

        while(<<now>> - current_timestamp < SCHEDULE_PERIOD); // Wait
    }
}
```

Figure 4.14 C++-like representation of automatic scenario.

How the code may notice, the thread that manages the automatic scenario presents a function called “is_runnable()”. This part of the automatic scenario is configurable by the configurators in order to define custom behaviours according the status of the whole system.

According the implementation of this function, automatic scenarios could belong to two different branches:

Pure periodic tasks: tasks are executed according the only scheduling. It can be used to fire specific procedures periodically. The code could be edited removing the runnable condition (“*is_runnable()*” return always *true*)

```
void periodic_scenario(){

    while (true){
        long current_timestamp = <<now>>;

        scenario();

        while(<<now>> - current_timestamp < SCHEDULE_PERIOD); // Wait
    }
}
```

Figure 4.15 C++-like representation of pure periodic scenario

Cron jobs: tasks are executed according the moment in which they are fired. In order to deploy a cron job, configurators must define a custom activation condition according the requirement. This kind of tasks are useful for deploying automatic procedures that does not matter about the plant status (e.g. turn on lamps at 8 p.m.).

Event driven tasks or **Smart scenarios**: tasks are executed according the status of the system/subsystem to which they interface. The activation condition gets the status at the beginning of each scenario fire and then IDOM performs the scenario if and only if this condition is satisfied. Smart scenarios are useful in industrial plants because they permit to control the status of the whole system and execute the required tasks, e.g. controlling an engine according its current speed or a heater activation according the external temperature.

The development of scenarios is possible according specific interfaces that allow configurators to define all step of the procedure.

4.1.5.3 Advanced scenario utilization

The great configurability of scenarios permits to deploy advanced procedures.

This section is exploited to illustrate a few examples of these advanced implementations.

4.1.5.3.1 Advanced utilization #1: local events

As introduced in the previous chapter, IDOM permits to configure virtual/local devices and events that live only inside the application server.

These local data could be useful to create interactions among users, devices and scenarios, e.g. maintenance procedures or enabling special UIs interactions.

4.1.5.3.2 Advanced utilization #2: parametric scenarios

Parametric scenarios represent special purpose user command scenarios that requires parameters from the caller. Due to their nature, only third-party application can launch them through web services.

The C++-like code could be modified by inserting parameters in the signature:

```
void parametric_scenario(string arg1, string arg2, ...){
    .
    .
    .
}
```

Figure 4.16 C++-like representation of parametric scenarios

4.1.5.3.3 Advanced utilization #3: Final State Machine or FSM configuration

IDOM smart scenario are useful to deploy automatic controls that manage the controlled processes, however they could be used to deploy FSMs.

An FSM could be implemented by the configuration of a mix of event driver and manual scenarios that can represent the transition functions from each state.

It is required to develop correctly the activation conditions of each automatic scenario in order to obtain the correct behaviour of the FSM.

This advanced functionality permits to reduce the complexity of the installed program in devices of the connected systems to leave it to IDOM server, configurable and extensible according the application needs, if there are critical issues on developments or there are requirements that could not be manageable internally to devices.

4.1.6 External services for notifying and controlling

IDOM functionalities permit to supervise and control systems according the methodologies explained in this chapter.

In most of the cases, customers could require to be advised about the status of the monitored system and, sometimes, they should send commands on it.

IDOM natively provides these functionalities using an extended monitoring and controlling engine that interfaces with the modules of the products.

The functionalities permit to configure custom messages according that will be sent according user configurations over the following services:

- Email
- Telegram
- SMS

4.1.7 Preventive Maintenance Engine

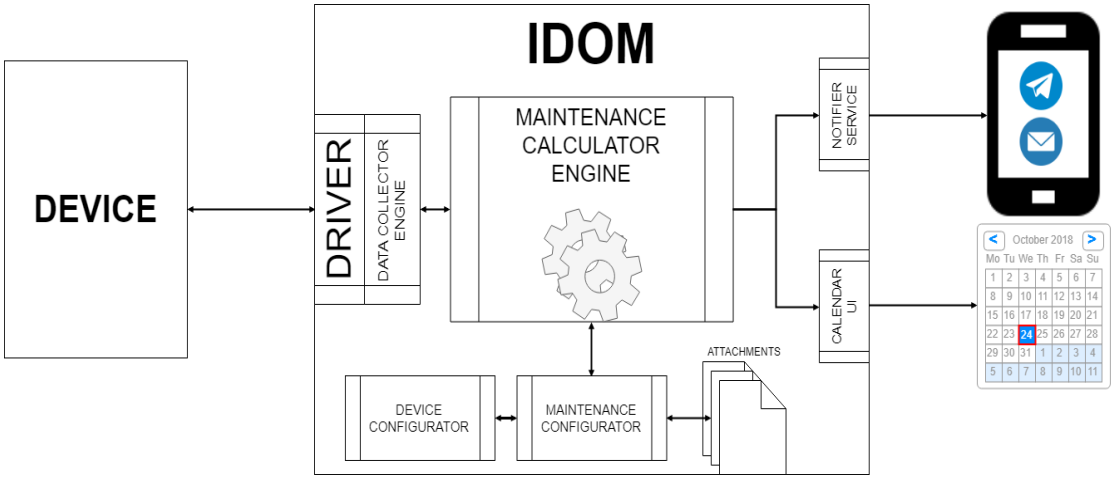

Figure 4.17 Preventive Maintenance Engine diagram

The SCADA application integrates a module that allows to calculate and predict maintenance events in the manner to organize the company schedules and tasks.

The engine is an owner module that allows to make predictions and projections with the following characteristics:

- Precise projections: projections could be deterministically calculated through the configuration of a wide range of selectable criteria that cover the most of all possibilities
- Flexible projections: projections depend on devices states and their utilizations. More the device is used, projections will be more frequent.

- Independent projections: it is possible configuring multiple and independent maintenance plans for each device of the plant that design instruments, replacements and staff that are interested to the intervention.

The engine module could collaborate with the others in order to permit advanced functionalities and interactions. The main example is the interaction between the maintenance engine and the notify one. The diagram illustrates how the modules are connected.

4.1.8 Video Streaming and Motion Detection Engine

The SCADA system provides an engine that permits the elaboration of video streaming through web sockets.

The module permits to develop the following application and functionalities:

- Video re-streaming

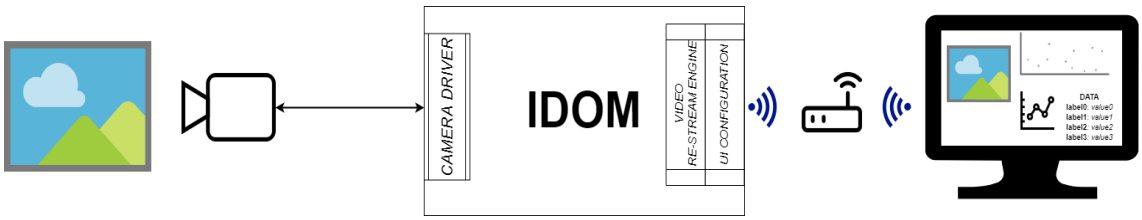

Figure 4.18 Video re-streaming diagram

- Video wall

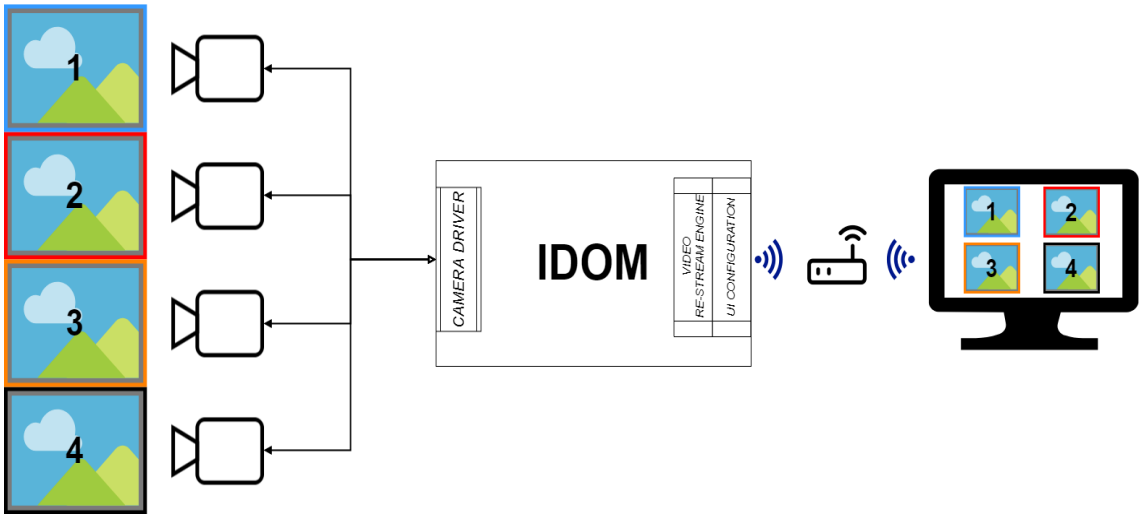

Figure 4.19 Video wall diagram

- Motion detection

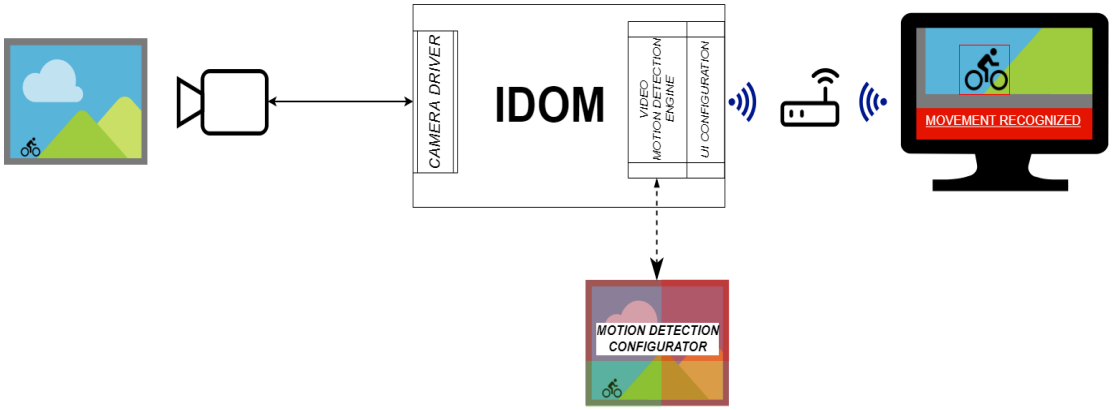

Figure 4.20 Motion detection diagram

The management of video contents is achieved through the management of sockets that can be redirect to any clients.

As it is possible seeing in the figure 4.20, IDOM provides an efficient internal motion detection engine. It is developed using OpenCV library. It is a famous open source library that is available in Internet for different programming languages.

4.1.9 Geolocation Engine

A common IDOM instance provides drivers and services that allows the application to collect data from devices and store events according the only time location as the unique metadata, but the system could collect also geolocation information.

The geolocation is fundamental to deploy applications that require to track the position of vehicles, e.g. trains.

Due to the nature of this kind of applications, IDOM provides drivers that work in two different modes:

- Real-time mode: data are collected according pseudo-real time policy. This fact implies events are geolocated on elaboration time and any connection loss mean also data loss
- Buffered mode: data are managed according policies that permit to recover any information that were not fetched previously. It implies that devices must hold all events with date and geolocation metadata.

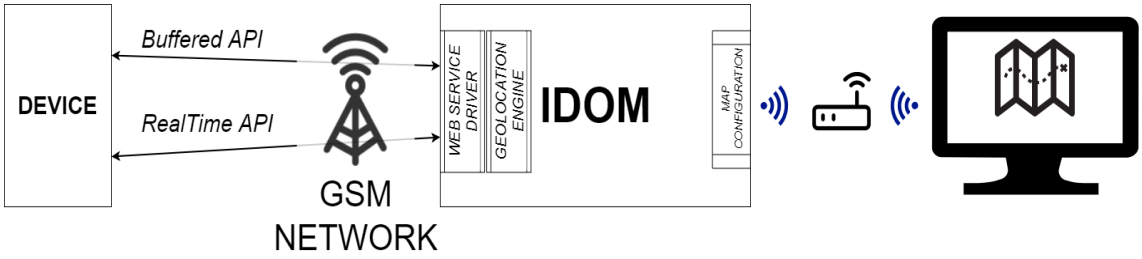

Figure 4.21 Geolocation communication through GSM network

Diagram of geolocation IDOM application over GSM network, e.g. trains tracking.

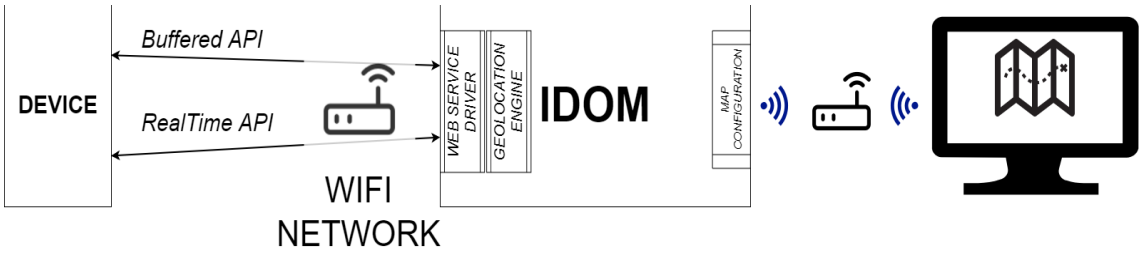

Figure 4.22 Geolocation communication through Wifi network

Diagram of geolocation IDOM application over Wireless network, e.g. telemetry for racing vehicle.

4.1.10 Expandable using web services

IDOM is an application that deploys all functionalities of a SCADA system, however customers could require special purpose functionalities that are extremely complex to develop with the built-in configurations and scenarios.

For this reason, IDOM exposes a set of web interfaces that permits integrate it as a module of a complex system (examples are illustrated later in the following chapter).

Available operations permit third-party application to perform:

- Reading data from IDOM or directly from devices
- Writing data to devices (if the operation is permitted by the driver/configuration)
- Execute custom queries in order to fetch data for the database
- Launch scenarios or parametric scenarios
- Show dashboards and synoptics, useful for deploying monitors or TVs that show them

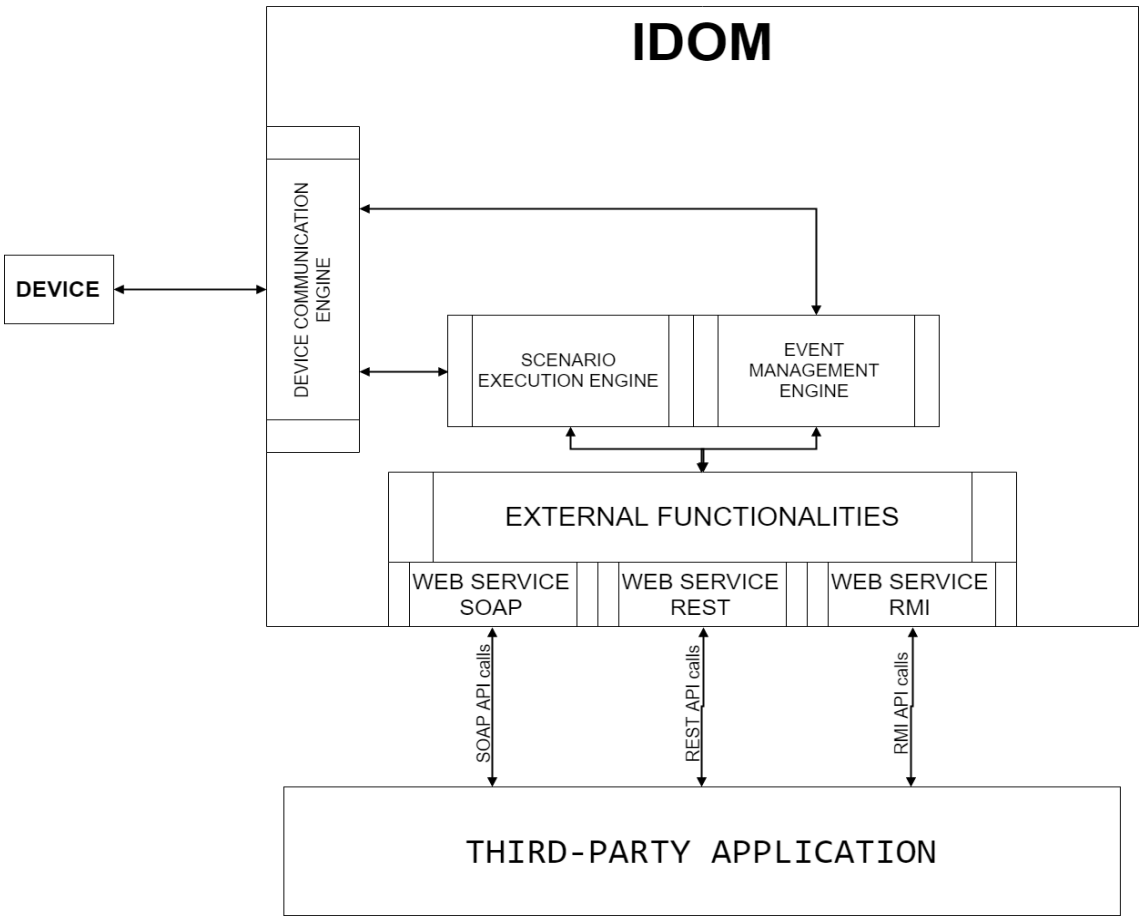

Figure 4.23 Third-Party application communication with IDOM

The diagram 4.23 shows how third-party applications could interacts with IDOM.

The available APIs are exposed using the following web interfaces in order to permit the development of applications without constraints about the interfacing:

- Representational State Transfer (REST) API web interfaces
- Simple Object Access Protocol (SOAP) API web interface
- Remote Method Invocation (RMI) API web interfaces

4.1.11 Responsive User Interfaces Configurator

As said at the beginning of the chapter, user interfaces of IDOM are responsive in the manner to be navigable any devices (workstations, smartphone, tablet, etc.) in which a web browser client is installed, like Google Chrome or Mozilla Firefox.

As said previously (section 4.1.1.2), UIs can be of the following types:

- Administration UI
- Back-End configuration UI
- Front-End configuration UI
- Dashboards and synoptics

IDOM provides a UI editor that allows the designing of graphical components that will be used to show the status of the controlled process, plants or, generally, paired systems.

The library of graphical components is composed of the following elements:

- Line Charts

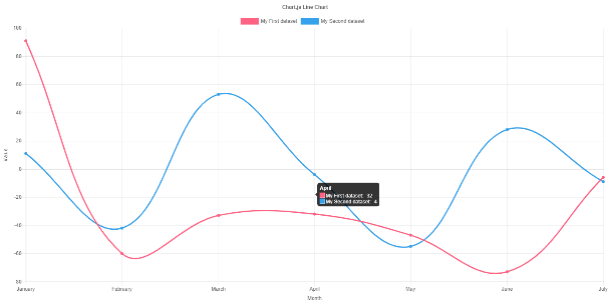

Figure 4.24

- Bar Charts

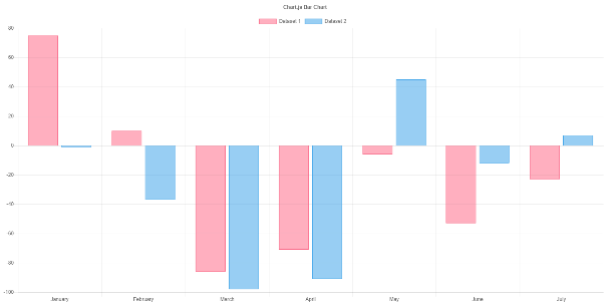

Figure 4.25

- Pie Charts

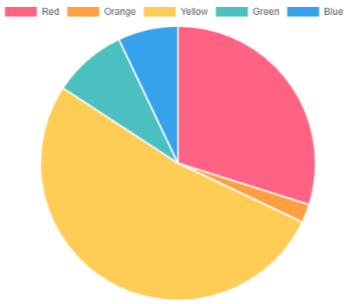

Figure 4.26

- Radar Chars

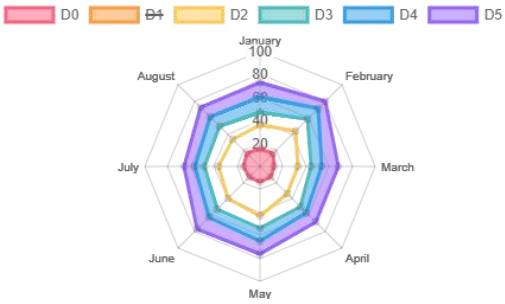

Figure 4.27

- Tables
- Static or dynamic HTML 5 output elements (labels and images)
- Static or dynamic HTML 5 output elements (labels and images)
- Configurable Canvas Gauges

Figure 4.28

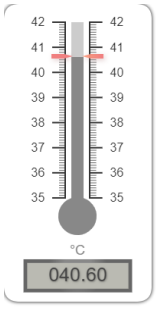

Figure 4.29

- Camera video contents

Figure 4.30

- Iframes
- Maps

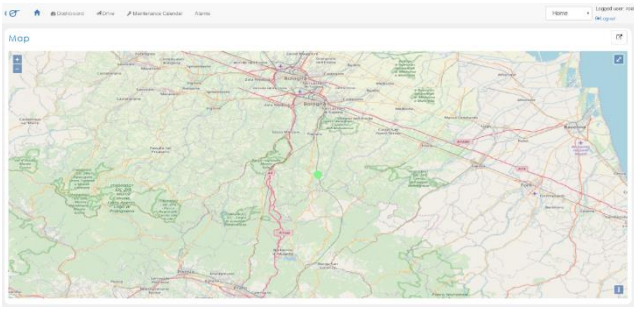

Figure 4.31

4.1.12 Embeddable

As described previously, IDOM is a modular application that can be distributed in different configurations according their major components and their utilizations:

- IDOM Core: the core of the application that performs the main functionalities of the SCADA
- Data Base server (MySQL Server) that stores configurations and historical data

These components can be installed on the same machine or in separated ones according the required performance of each of them.

According the adopted configurations and the required performance of IDOM, it may be installed in two different versions this defined:

- LOGGER: it is the version that provides the interaction with the plant for monitoring and acquiring diagnostic/production data and alarms in near real-time. By its nature, the LOGGER system has a limited storage memory. Before this amount of memory will be saturated, a data aging policy is applied.
- BIG DATA: it is the version for logging a large amount of data that are subjected to calculation logics relating to preventive maintenances and parametric analysing of events occurred on plants. By its nature, the system requires high HW specifications to work as expected.

4.1.12.1 LOGGER installation

LOGGER application is the component of IDOM that communicates directly to the plant and it has the task to make available interfaces to web clients who connect to it.

LOGGER can be configured in 2 different modes:

- ON-BOARD MACHINE installation
- INTRANET installation

4.1.12.1.1 ON-BOARD MACHINE installation

ON-BOARD MACHINE installation means the possibility to integrate the application directly in the monitored/controlled device through a reduced dimensions and performance industrial pc that is used to collect a limited number of events.

HW features of the computer define the abilities below:

- Elaboration/Acquisition performance
- Storing memory volume
- Number of associable devices
- Number of supported clients/sessions

4.1.12.1.2 INTRANET installation

INTRANET installation means the possibility to install the application on a local server that communicates to monitored/controlled machines through the local network.

The server could be:

- Pre-existing server with installation in the physical machine
- Pre-existing server with installation in a virtual machine
- Server provided by ERInformatica S.r.L.

This application version could manage multi-devices connections thanks its HW features that depend on volume of data to be processed and stored.

4.1.12.2 BIG DATA installation

BIG DATA application provides the logging of all events recorded by LOGGERs.

BIG DATA could be configured in 2 different ways:

- Embedded installation into the LOGGER
- Installation on a server

4.1.12.2.1 Embedded installation into the LOGGER

This configuration allows to integrate in the same machine both of LOGGER and BIG DATA components.

Due to the configuration nature, the installation must be performed on a local server (INTRANET installation) with adequate performance that can be parametrized based on the data volume to be historicized and processed.

4.1.12.2.2 Installation on server

The application is installed on a dedicated server which receives the data collected by the associated LOGGERs via the web service.

The server set up for BIG DATA can be indistinctly installed in one of the following ways:

- Intranet: the server is connected to the customer's local network for which the reachability is guaranteed from the IDOM LOGGER instances.
- Cloud: the server is a remote one which is available through internet connection for which the reachability is guaranteed from the IDOM LOGGER instances

4.1.12.2.3 Modular installation

Modular installation means the possibility of breaking down the application components in separate machines, each of which contains at the least one of the following modules:

- Apache Tomcat 8
- MySQL Server

This configuration makes flexible choosing the machine HW according its use in IDOM.

Each module respects the installation modes presented previously.

4.1.12.3 Overview Diagrams

The diagrams below show the available configurations of the IDOM application in terms of network architecture and HW requirements.

4.1.12.3.1 LOGGER: ON-BOARD MACHINE; BIG DATA: INTRANET

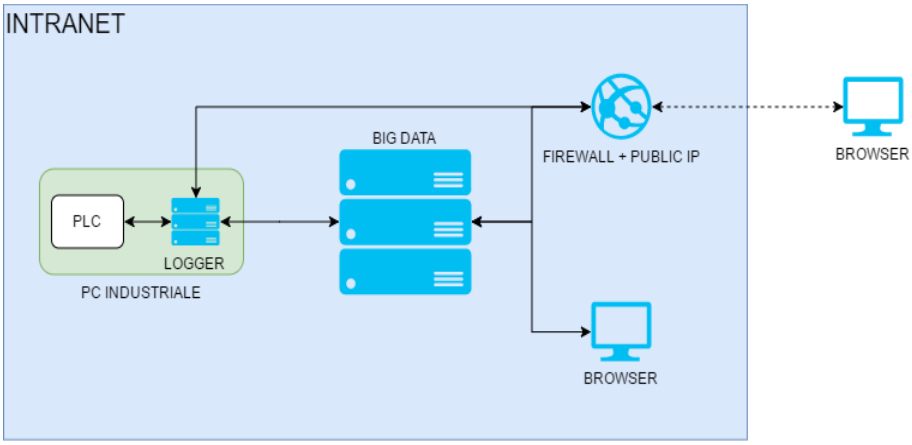

Figure 4.32 Diagram of On-Board Machine Logger and Intranet Big Data

PROs:

- In-house management of the Big Data server
- Reduced cost due to ad hoc Logger HW features according to the associated PLC

CONS:

- HW and systemic maintenance levy on the customer of the Big Data server
- Networking configuration costs to make services available on internet
- Higher costs according the number of associated PLC

4.1.12.3.2 LOGGER: ON-BOARD MACHINE; BIG DATA: INTERNET

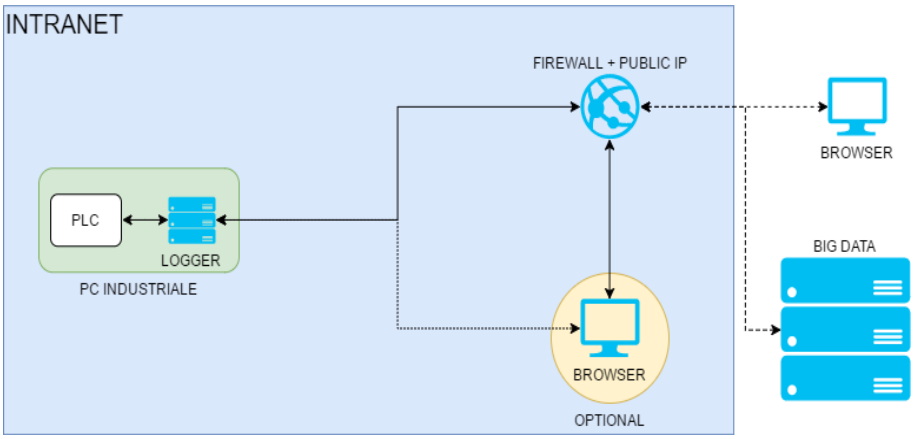

Figure 4.33 Diagram of On-Board Machine Logger and Internet Big Data

PROS:

- Unique access through web interfaces to collected data through distinct production plants
- Reduced cost due to ad hoc Logger HW features according to the associated PLC
- Logger and Big Data components physically separated
- Optional Logger web interface (Logger UI configuration Lite)

CONS:

- Higher cost according to the number of associated PLCs

4.1.12.3.3 LOGGER: INTRANET; BIG DATA: INTEGRATED

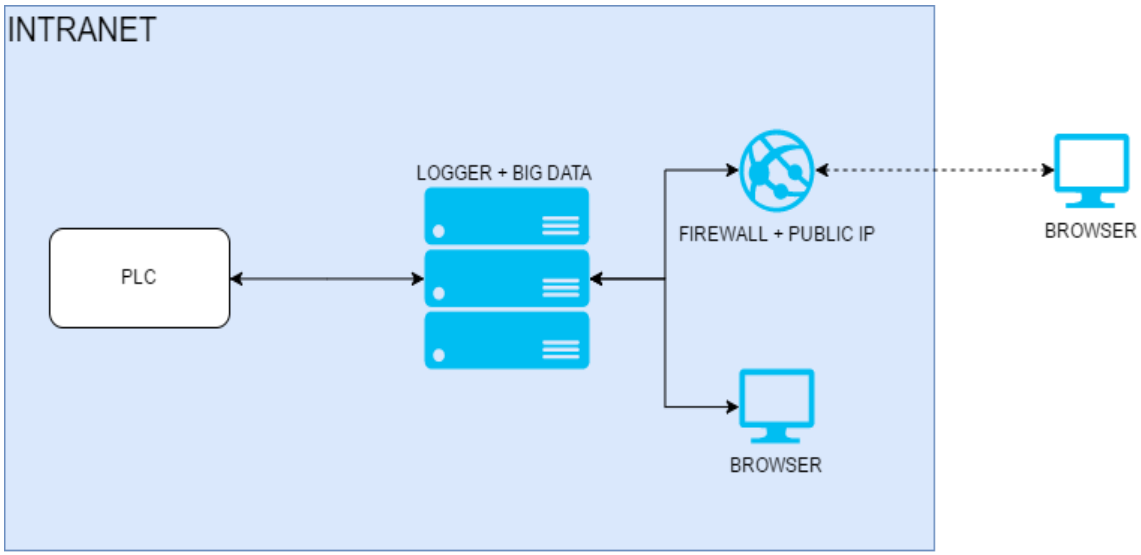

Figure 4.34 Diagram of Integrated Logger and Big Data

PROS:

- In-house server management
- Logger HW features adequate for management of greater number of PLCs

CONS:

- Higher infrastructure cost
- HW and systemic maintenance levy on the customer
- Networking configuration costs to make services available on internet

4.1.12.3.4 LOGGER: INTRANET; BIG DATA: INTRANET

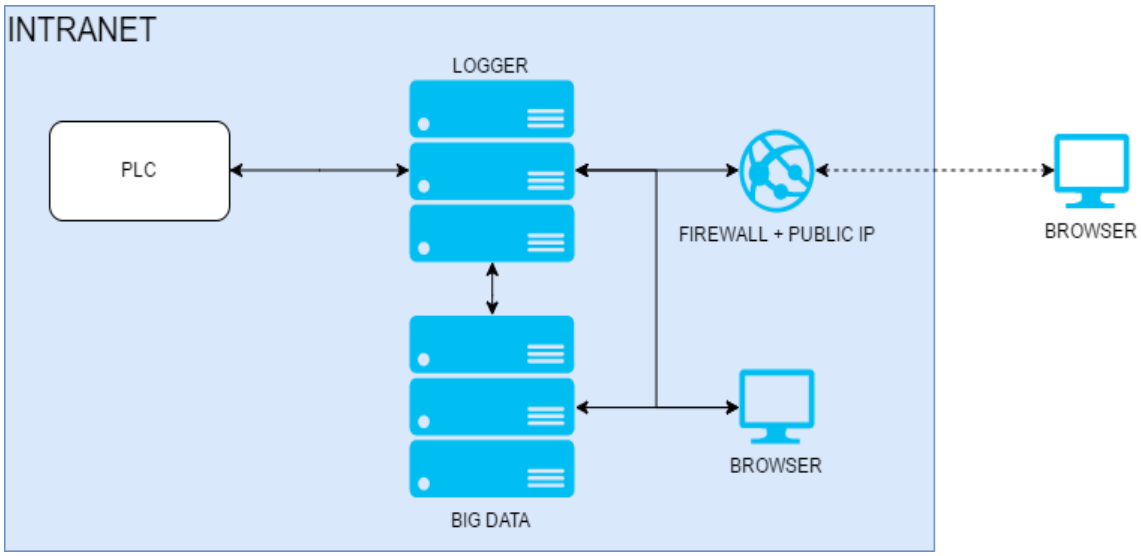

Figure 4.35 Diagram of Intranet Logger and Big Data

PROS:

- In-house management of the servers
- Logger HW features adequate for management of greater number of PLCs
- Logger and Big Data components physically separated

CONS:

- Higher infrastructure cost
- HW and systemic maintenance levy on the customer
- Networking configuration costs to make services available on internet

4.1.12.3.5 LOGGER: INTRANET; BIG DATA: INTERNET

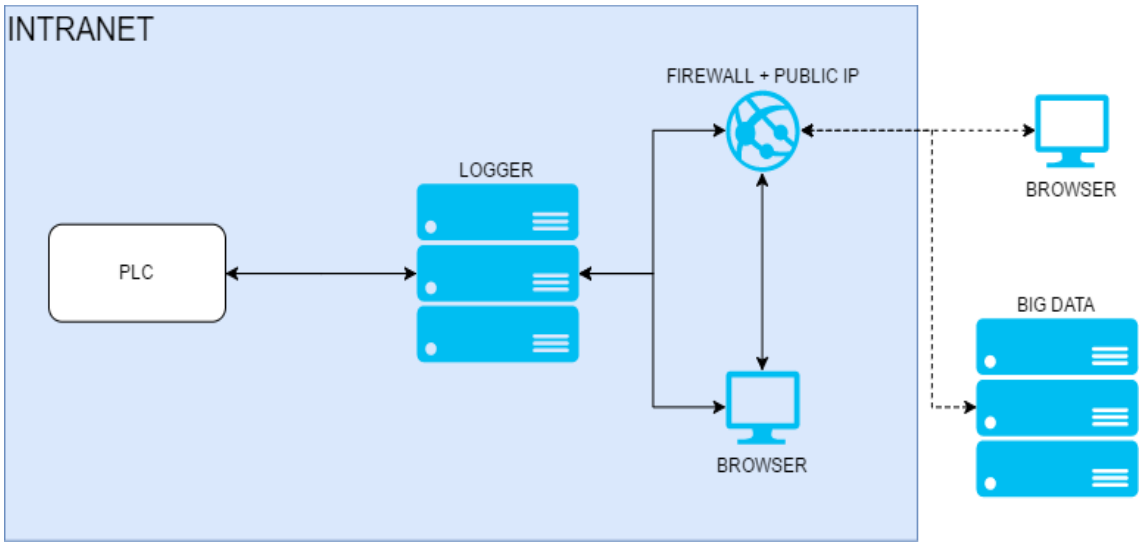

Figure 4.36 Diagram of Intranet Logger and Internet Big Data

PROS:

- In-house management of the Logger server
- Unique access through web interfaces to collected data through distinct production plants
- Logger HW features adequate for management of greater number of PLCs
- Logger and Big Data components physically separated
- Optional Logger web interface

CONS:

- Higher infrastructure cost
- HW and systemic maintenance levy on the customer

4.2 STRENGTH OF IDOM

This section is exploited to describes the main features which composed IDOM.

4.2.1 Configurability

IDOM system is heavily configurability in their components and entities in order to achieve the needs of heterogeneous application fields in which it may be integrated.

The main configurable components are:

- CRM entities:
  - Societies
  - Users
  - Grants
- Data acquisition entities:
  - Event categories
  - Devices
  - Drivers
- Web app module:
  - Dashboards
  - Synoptic
  - Graphical components
- Programmable logics or Scenarios:
  - User or Manual commands
  - Automatic or Smart scenarios

4.2.2 Supported devices and Data Sources

According the IDOM point of view, devices act as generic data sources or controllable modules that communicate with the SCADA through the exchanging of messages over the network infrastructure.

To date, IDOM is provided of communication drivers that allows the interaction with the following "*device*" families:

- Programmable Logic Controllers (PLCs)
  - Ethernet IP
  - ModBus
  - ProfiBus
  - OPC UA
- IP Cameras:
  - Insight cams
  - DataMan 2000 cams
  - Web cams
- Relational DataBase Management Systems or RDBMS

    - MySQL
    - SQL Server
    - DB2
    - DB2 iSeries – AS400
- CSV and Electronic Worksheets (e.g. *xls* files):
    - Local filesystem
    - FTP server
    - SAMBA or CIFS shared folder
- Virtual/local devices
- Web services
    - SOAP API
    - REST API

5 CASES OF STUDY

5.1 SUPERVISION – CALDIROLA

Supervision is the application of IDOM that implies only functionalities of supervision.

Caldirola requires to develop a supervision system for its establishment of Missaglia that allows the monitoring of parameters, listed below, of a stocking area of sparkling wine:

- Pressure
- Temperature
- Wine Level
- Glycol supply temperature
- Glycol return temperature

The customer requires the possibility of sent mails to specific users when parametric conditions are satisfied.

The integration of IDOM in the establishment and the necessity to speed up productivity requested the replacement of the whole previous IT infrastructure that was responsible to data fetching and HMIs due to it was incompatible with the new implementation: the old system did not allow to reach the maximum production speed.

5.1.1 Implementation

IDOM is used to the following purposes:

- Sampling data from sensors through TCP/IP network bus
- Deploying user interfaces for analyse the collected data for short/medium time periods

5.1.2 Benefits

- Installation and test time of the supervisory system was extremely reduced due to no addons or developments were required
- The flexibility of IDOM permits a rapid configuration of alarm conditions and notifications

5.1.3 Future developments

The customer is satisfied of the implemented solution, so it is expected the configuration of further sensors

5.2 MARTINI & ROSSI – TECHNOLOGY BSA S.P.A

The new export regulations for wine products to Russia requires a precisely bottle tracking, which are identified by unique state marks provided by the Russian government. The manufacturer must perform the tracking also for cases, pallets, bottle positions in cases and case positions in pallets.

For this reason, ERInformatica S.r.L. develops a custom application that integrates IDOM as intermediary among devices and software system that was installed into two production lines called L24 and L42.

The requirement to reach a greater productivity, the previous IT infrastructure was replaced with an upgraded one that permits to reach the maximum speed of the production lines.

5.2.1 Implementation

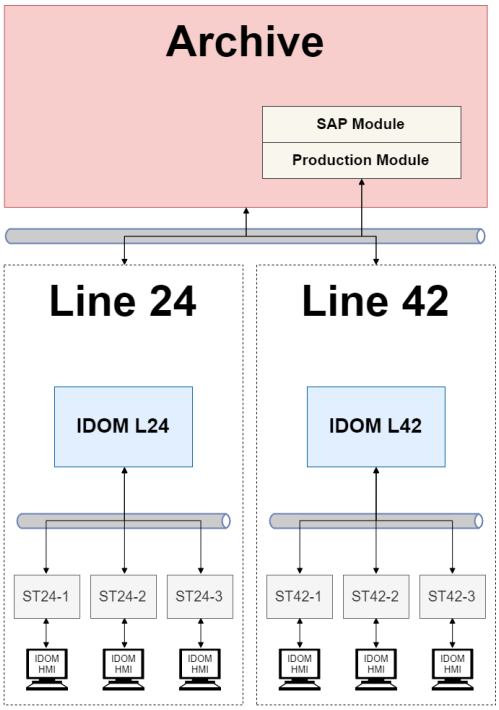

Figure 5.1 Martini & Rossi block diagram

The diagram shows the adopted solution architecture.

IDOM tasks:

- Retrieving the production order from ARCHIVE module, with the settings of the plants (e.g. the used data mark reel)
- Interacting with line devices (labelling station, boxing station and pallet station)
    - Production recipes
    - Counters and other data logging
    - Assignment of marks to boxes

    - Assignment of boxes to pallets
- Providing on board HMIs over machine touch panels
- Re-stream video contents from line cameras to HMIs

ARCHIVE module tasks:

- Communication with SAP system
- Acting as intermediary between SAP and IDOM
- Joining collected data from IDOM and elaborating shipping reports

5.2.2 Benefits

- The developed system respect widely in terms of performance and data quality according the customer requirements and needs
- It was possible implements easily the differences among the two production lines using the same architecture
- Eventually updates of memory locations of controlled PLCs can be easily integrated in IDOM due to its flexibility without re-distributing a new version of the application
- The adopted solution permits to centralize modules that previously was distributed (HMIs, video streaming, …).

5.2.3 Future developments

The customer is satisfied of the adopted solution, so it is expected the integration of the system in a third production line of a different establishment.

5.3 MIMESIS – LOGICO S.R.L. / INFINITY BOTTLING

The customer LogiCo had expressed the need to develop a Manufacturing Execution System (MES) for the management of production lines and for the elaboration of statistics relating to productions carried out or in progress to monitor the efficiency of the same based on the events recorded by IDOM module.

The mathematical model for statistic calculations is currently based on:

- Supervision: system module that involves one or more IDOM instances that collect data from machines
- LogModule: system module that implements the functionalities of:
    - Elaboration of data collected by IDOMs
    - Management of articles, customers and logistic units
    - Calculation of efficiency of productions based on the mathematical model:
        - # produced pieces / # total pieces
        - Utilization time (UP TIME) / total time (AVAILABLE TIME)
    - Configuration of production recipes through web user interfaces or CSV importing (only for line technicians)
    - Creation, launch and analysis of production orders
    - Notification engine through e-mail or Telegram (only for line technicians)

Recipes represent the available configurations of the production lines and the involved machines. They are identified by:

- Recipe name
- Recipe type (ACTIVATION or DEACTIVATION)
- Last device (usually PLC) of the configuration that represent the MASTER of the recipes
- List of monitored events
- List of related recipes for STOP, PAUSE, RESUME the production batches
- List of IDOM scenarios that will be send at the time of recipe launch
- List of pairs Article/Customer related to the recipe, if this option is enabled
- List of printers to configures when the order starts

5.3.1 Implementation

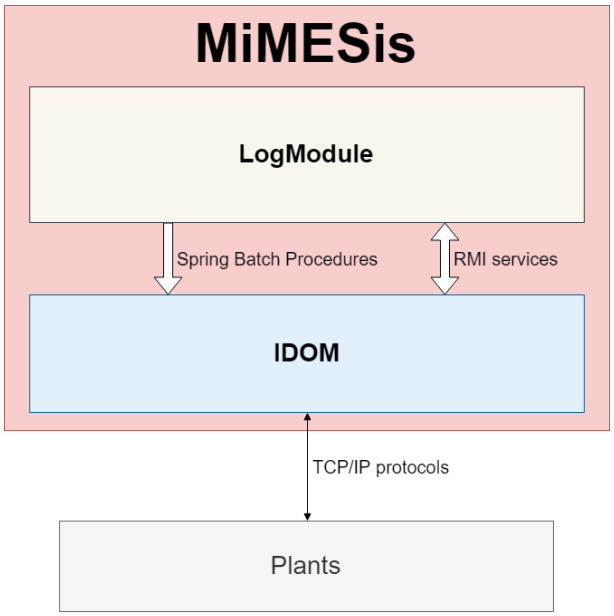

Figure 5.2 MiMESis block diagram

The described system is based on the installation and configuration of three types of actors:

- I.D.O.M. servers for the supervision and data collection. They are the core of the MES system
- LogModule server for elaboration of collected data by supervision for the management of the whole system
- PLCs in the production line (their configuration is demanded to LogiCo according their issues)

5.3.1.1.1 IDOM – Machine interactions

The communication between IDOM instances and line machines is implemented according to a communication protocol defined in agreement with the customer.

Currently it consists of an FSM which, through the exchange of messages, advances in its internal states.

The protocol is natively developed in IDOMs through the configuration of automatic scenarios (see 4.1.5 for more details).

The exchanges of messages it is deployed using two simplex communication channels that interact with two different memory locations in the controlled devices:

- *Send/FROM_PLC*: memory location that is written by PLC and read by IDOM
- *Receive/TO_PLC*: memory location that is written by IDOM and read by PLC

5.3.1.1.2 LogModule – IDOM interaction

The LogModule module implements the following functionalities that permit the interaction with IDOMs:

- Spring Batch procedure, manually launched by administrators, that permits to retrieve the entities configured in IDOM instances:
  - Devices
  - Event categories
  - Scenarios
- Spring Batch procedure, scheduled and parametrizable by administrators, that permits to fetch collected events from IDOM instances
- RMI web interfaces for starting scenarios of IDOM instances when a recipe is launched, eventually with extra information e.g. target produced pieces or article description
- Web UIs that are accessible using their URLs for the visualization of dashboard, synoptics and customizable kiosks

5.3.2 Benefits

- Parameterization and configurability of lines based on the used recipes
- Supervision of the line and automatic start of scenarios for the management of the same
- Collection of information related to production orders
- Standardization of line machine communication interfaces – supervision system

5.3.3 Future developments

MIMESIS was born as prototype and start point of a more complex and sophisticated application.

The developments and improvements could be summarized in:

- Applying the Overall Equipment Effectiveness (OEE) mathematical model
- Improving the management of production recipes
- Improving the management of production orders
- Improvements of devices in industrial plants
- Improvements of kiosks for data visualization
- Definition of a standard interface that permits to establish regulations about the development of machines in terms of communication protocol/interface

6 CONTRIBUTION

The contribution given by my work permits to accelerate the development and configuration of many modules of IDOM and third-party applications that interact with it.

At the beginning of the development, I had to learn the whole technology stack in order to design correctly the assigned components:

- Spring Core
- ZK 7 Framework
- JPA and Hibernate Implementation
- Documentation about drivers of different PLC vendors

After the training period, my tasks were identified in 3 different groups:

1. Development and configuration of communication drivers
2. Designing and implementation of several UI functionalities
3. Configuration of IDOM and third-party application

6.1 DEVELOPMENT AND CONFIGURATION OF COMMUNICATION DRIVERS

As described in this paper, IDOM owns the capability to communicate with several real or virtual devices. This feature makes the system easily integrable in a huge amount of plants without creating constraint to plant designers.

The first tasks that were assigned to me concerned the designing of the first version of data structure and driver interfaces in collaboration of my colleague and supervisor Eng. De Sisto.

We analysed different solutions in order to exploit benefits and malus of them and, at the end, obtain the most flexible data model and interfaces that are illustrated in this document.

The next task was to implement and configure some drivers to integrate in IDOM. The implementations that I developed allow to integrate into the SCADA the following “device” families:

- Databases over JDBC driver (MySQL, DB2, SQL Server)
- Static files in local filesystem or in shared folders (FTP or CIFS)
- OPC UA communication protocol
- Web service drivers

The last task that I accomplished was the integration of utilities and functionalities that allow the geolocation of events and devices using OpenLayer.

6.2 DESIGNING AND IMPLEMENTATION OF SEVERAL UI FUNCTIONALITIES

Due to the low number of dedicated employees to the project, I had to carry out tasks that concerned also the implementation of graphical user interfaces and functionalities in IDOM. During this step the main task consisted of the integration of graphical libraries for charts and tables using ZK and Chart JS facilities.

The integration of graphical components required to carry out research in order to find the most useful and efficient solution according which libraries were to be included and how install them.

6.3 CONFIGURATION OF IDOM AND THIRD-PARTY APPLICATION

At the end of the development of IDOM core, it was requested to install the SCADA application in several plants.

As described before, some utilizations contemplate the integration of specific configurations and external components that accomplish the specific behaviours that IDOM does not provide natively.

My developments act to provide the functionalities and tasks that communicate directly with IDOM and, therefore, the devices configured in the SCADA system.

This fact allowed me to meet professionals and confront them according strategies and tips in the development and analysis phase of the integration of the SCADA or plants, especially during the development of Martini Rossi project, in which I developed entirely the module that interacts with IDOM.

Further the previous project, my supervisor and my employer gave me the responsibility to occupy of the configuration and inspection of the application MiMESis for Infinity Bottling company in California and, actually, the customized version of it for Moet Chandon company in Argentina.

6.4 MY ACTUAL ROLE IN THE PROJECT

Currently, my role in the company covers activities that consist on maintenance, configuration and integration of IDOM in plants in which it will be installed.

In addition to this, my tasks include the designing and development of newer functionalities and features.

7 CONCLUSION AND FURTHER DEVELOPMENTS

The aim of the author was to examine and discuss the potentiality of SCADA systems and describe their utilizations considering the product developed by the company ERInformatica S.r.L. called Integrated Devices Open Management or IDOM.

How it was described at the beginning of the paper, SCADA systems have been relevant since the beginning of the 50s due to the necessity to increase the productivity of companies and factories. They are still evolving to improve the provided functionalities and benefits that characterized this kind of systems, especially in reference to the new context of Industry 4.0 and Smart Factories.

All prefixed goals of the SCADA application were reached with excellent results and, for this reason, it is expected to develop a new version of the software system that realizes new features that permit IDOM to be more flexible and usable than now:

- Integration of an artificial intelligence that permits to execute functionalities of pattern recognition and forecasting about alarms, breaks and performance of industrial processes with the implementation of machine learning strategies
- Expansion and improvement of the driver library that will allow IDOM to interact with devices and data sources that are not pairable currently.
- Expansion graphical component library with the insertion of new elements which will permit the creation of more sophisticated dashboards and synoptics
- Improving data management system in order to permit

The development of the system has emerged analysis related to today's technologies trend and realistically applicable to the next decade one.

Based on the current state of art of IDOM, which achieves the concept of Internet of Things environment, e.g. the plurality of protocols (ISO OSI application level) and communication buses (ISO OSI Communication Network and Transport), it is evident that a polymorphic and flexible system could reach a wide horizon of heterogeneous sectors and utilizations, e.g. IDOM could be used as SCADA, video content analyser or preventive maintenance calculation module.

The commercial and real feedback of the developed product highlights a real and concrete interest in a SCADA that can be easily integrated with devices of different vendors and company information systems.

In conclusion, it is possible identifying more and more clearly in the adoption of design and development methods oriented not only for the re-use of components but also for the parameterization of the same the guideline applicable to the development of modern IT systems, especially for SCADA systems.

8 BIBLIOGRAPHY

[1] Bimbo Stefano, Colaiacovo Enrico, *Sistemi Scada – Supervisory Control And Data Acquisition*, Milano, Apogeo SRL, 2006

[2] Baotong Chen, Jiafu Wan, Lei Sshu, Peng Li, Mithun Mukherjee and Boxing Yin, *Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges*, IEEE Access, December 14, 2017

[3] Z. Ignaszak, R. Sika, M. Perzyk, A. Kochański and J. Kozłowski, *Effectiveness of SCADA Systems in Control of Green Sands Properties*, 2015

[4] ZKoss, *ZK Essentials*, http://books.zkoss.org/zkessentials-book

[5] ZKoss, *ZK Developers' Reference*, https://www.zkoss.org/documentation#References

[6] Mark Paluch, *Spring Vault - Reference Documentation*, 2017, https://docs.spring.io/spring-vault/docs/1.1.1.RELEASE/reference/html/

[7] Hibernate, *Hibernate Getting Started Guide*, http://docs.jboss.org/hibernate/orm/5.4/quickstart/html_single/

9 ACKNOWLEDGEMENTS

Desidero ringraziare tutti coloro che mi hanno sostenuto durante la stesura della tesi e il mio percorso universitario.

Tengo a ringraziare anzitutto il Professore Paolo Ernesto Prinetto che mi ha permesso di conseguire la tesi su un argomento da me proposto.

Ringrazio il CEO di ERInformatica S.r.L. Emilio Ravotti, il capo progetto l'Ingegnere Giovanni De Sisto e il collega Goran Postolov che mi hanno dato l'opportunità di partecipare al progetto IDOM e di entrare nell'azienda.

Ringrazio la mia famiglia che mi ha sostenuto durante l'intero percorso universitario, sia economicamente che moralmente.

Ringrazio la mia fidanzata Chiara che ha condiviso con me gli alti e i bassi del mio percorso di studi.